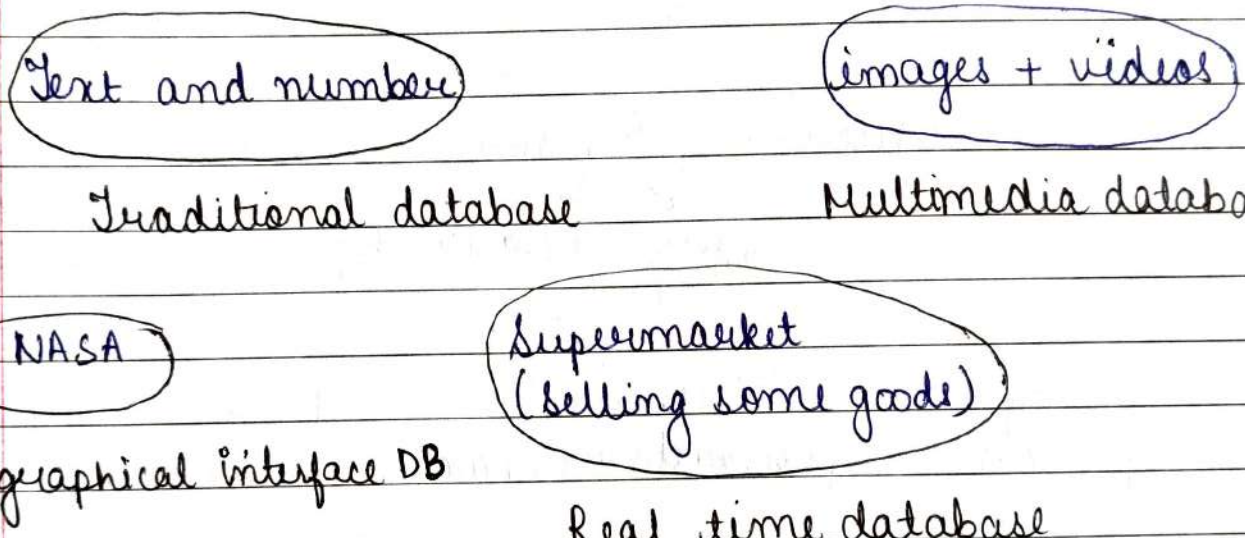
# Notes Of DBMS (SQL)

- By Riti Kumari

# Module 1 (ER model)
## Introduction to DBMS

Data - It could be any fact that can be recorded or stored.
Eg - text, number, images, videos, speech

Database - Collection of related data. (Only for text and number right now)




Data warehouse - The data is huge and historical
It is a kind of database.
Database mangament system - Set of programs used to defining datatypes structure & construct (place the data in the harddrive) data & manipulate it.

Types of DB
1) Traditional Database
2) Multimedia DB
3) Geographical interface DB.
4) Real time DB.

Date __/__/____

# DB models in Database.

| Modelling level | | |
|---|---|---|
| Level 1 | High level view or Conceptual view | (E-R model) |
| Level 2 | Representational or implementation. | (tables or ~~SQL queries~~ or Relations) |
| Level 3. | Low level or physical view. | (how to store, datatype etc) |

## Introduction of E-R model

Entity | Relationship

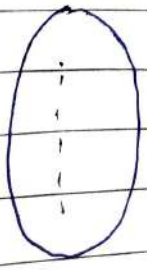
| Entity | Attributes | Relationship |
|---|---|---|
| A person | phone, address, name | owns a |
| Things | properties of entities | association among entities |
| Person | name, age, phone no | works for |

Entity type - Entity → (26, Raj, BKs)

↓ (Schema)

PERSON (Age, Name, add)

intension extension



state of database

Date ___ / ___ / _____

Schema - heading (how do you represent)

ER diagrams → Entity type.
↓
for a communication establishment.

ER diagrams
↓
Relational table
↓
Harddisk

DB attributes (to describe something)
↓
the more attributes the better information

Person → Name, age, address, phone no.
(Entity) (attribute)

Types of attributes

1) Simple and composite attribute
↓ ↓
can't be divided further — can be divided further (composed of many attributes)

Name
↓
F. Name — M. Name — Last Name

NULL - the value doesn't exists or not applicable.
Eg - Middle name

Date ___/___/_____

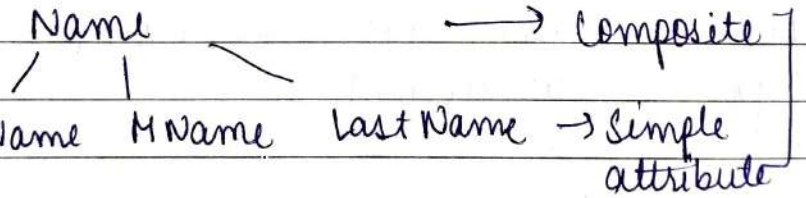
2) Single valued vs multivalued attribute

| Only one value | More than one value. |
|---|---|
| Eg - age. | Eg - address (permanent or residential address) |

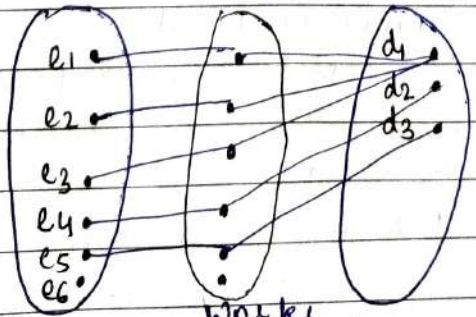
3) Stored vs derived attributes

| We store these values | Depending on the data we derive |
|---|---|
| Eg - DOB. | Eg - Age. (deriving from DOB) |

4) Complex attributes
Collec$^{n}$ of attributes or Comb$^{n}$ of attributes.




DB Relationships
1 TO Many




Requirement Analysis
Every employee works for a dep and a dep can have many emp. New dep need not have any emp.

Page No.

Date / /

Structural constraints

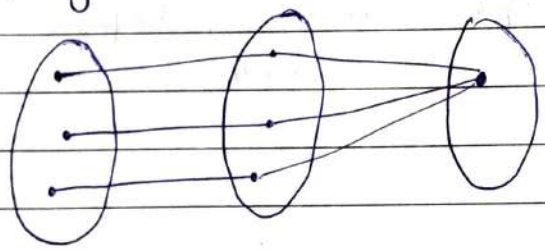
- Degree - (how many entity set is participating)

  Degree - 2 (Employee, department)

- Cardinality Ratio - what is the max^m no of relationship in which an entity can participate.

Cardinality 1 N

Employee Department

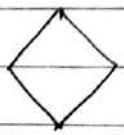


Participation or existence (Min^m cardinality)

Min^m relationship in which it can participate.

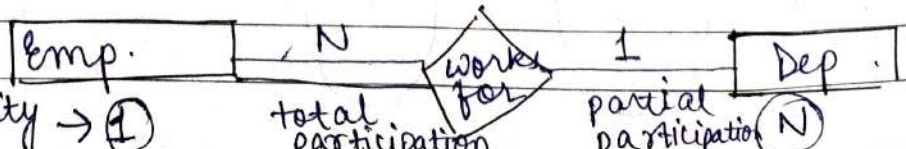
In case of Employee participation is 1.
New dep need not have any employee. Therefore participation is 0 in this case of dep.

Entities



Relationship

1 dep. can have many emp.



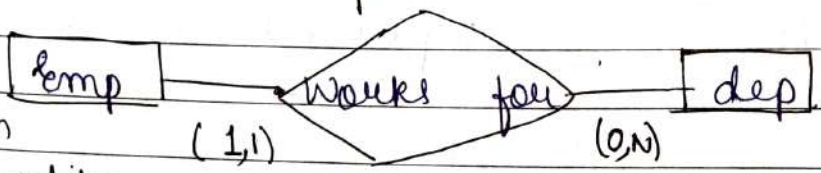
Date ___/___/_____

total participation
All entities are participating
═══

partial participation
If some entities are participating
———

Cardinality double line ration

Min^m max representation




Min^m Cardinality,
Max^m Cardinality

Min^m - max^m relation or representation elaborates more information

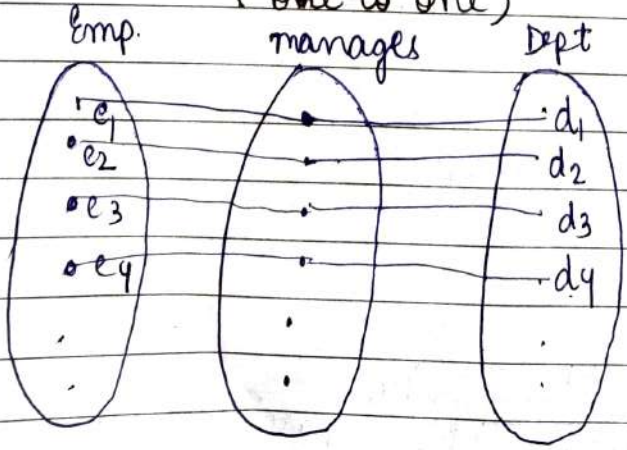
Role name

Employee

Employer

(Employee, manager) → role name

DB relationship (one to one)



Date / /

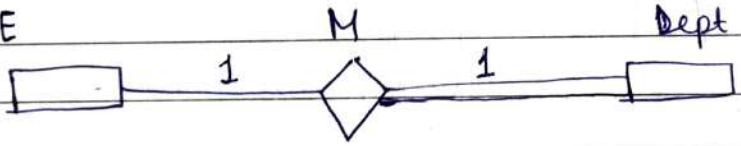
Every Dept should have a manager and only one employee manages a dept.

Any employee can manage only one department

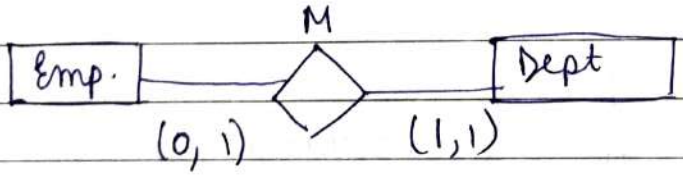
Degree → 2

Cardinality → { 1 (max^m no of relationship one entity can manage)
1

Participation → Employee (P) 0 (min^m no of rel^n.)
Dept (T) 1 (Every dept should have a manager)

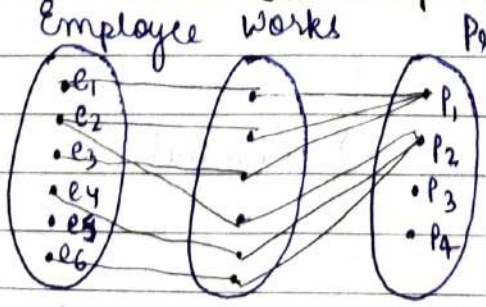



ouble line




in max

DB Relationship (Many to many)




$P_1$: $e_1, e_2, e_3$
$P_2$: $e_2, e_4, e_6$
$P_3$: $e_5$
$P_4$: $e_1, e_5$

Degree 2
Cardinality N
Participation 1

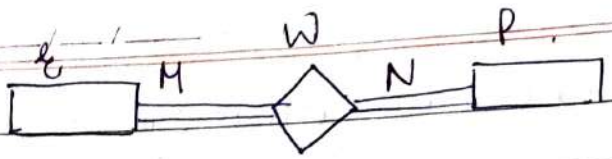
N
1

Req. Analysis - Every employee can work on atleast project & an employee can work on many projects

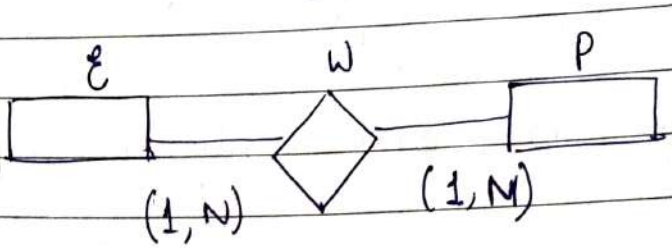
Page No.

Single L - Double line

Date __/__/__




Min - max Repr$^{n}$

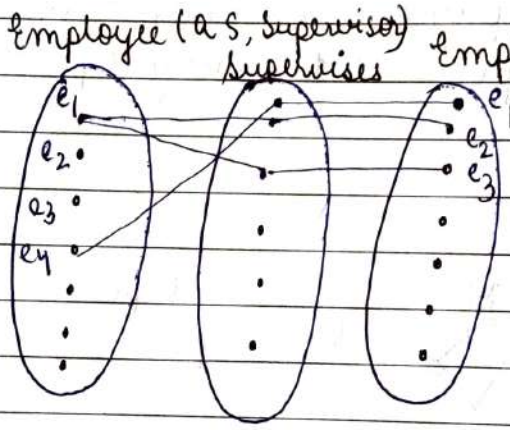


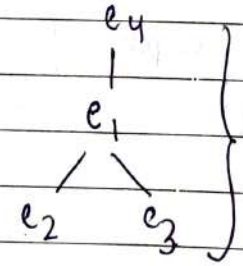

## DB recursive relationships

Even though its recursive its binary (2 entities only)

An employee managing other employee
(Boss) (secretary)

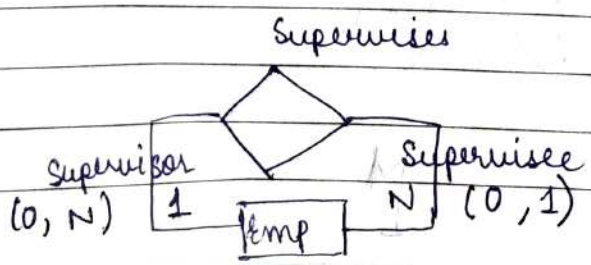
- as → acts as



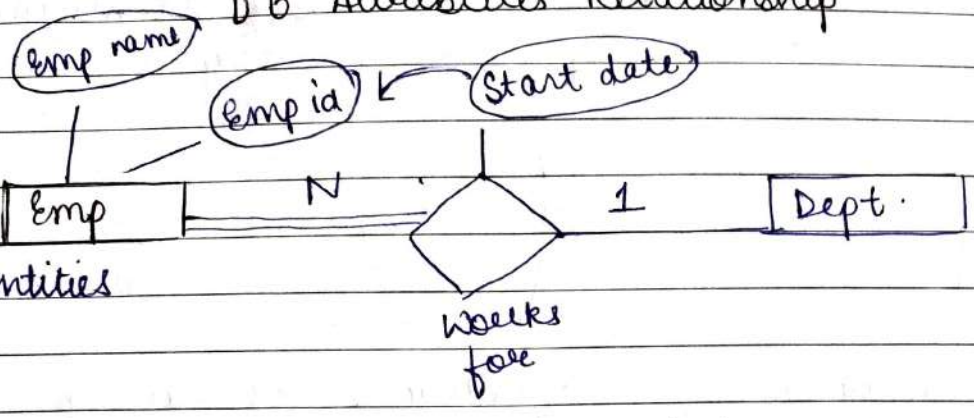

Recursive relationships

Degree : 2 entities
Cardinality : N 1
Participation : 0 0 (Partial P)



Date ___/___/_____

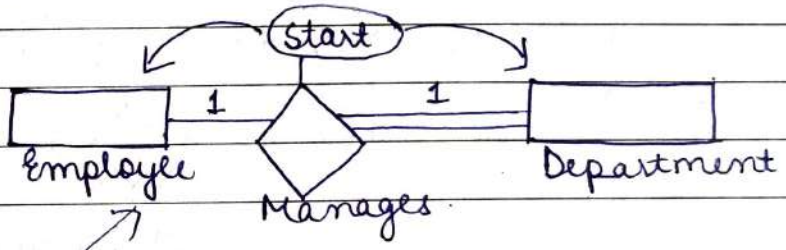
## DB Attributes Relationship




We want to have as less as attribute to the relationship

So its better to give the attribute at the n side or many side as every employee can have a start date.

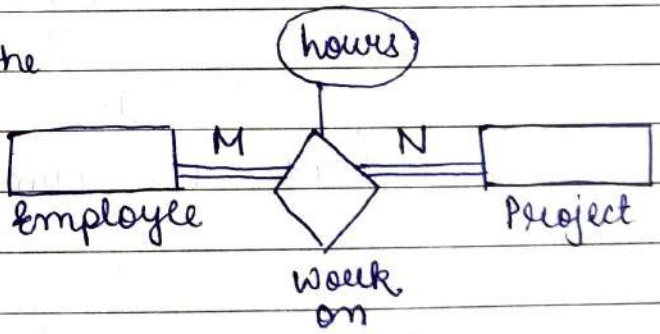
→ Many (N) ⟶ 1 (1)




Put the attribute to the N side

1 ⟶ 1

Any side we can move the attribute

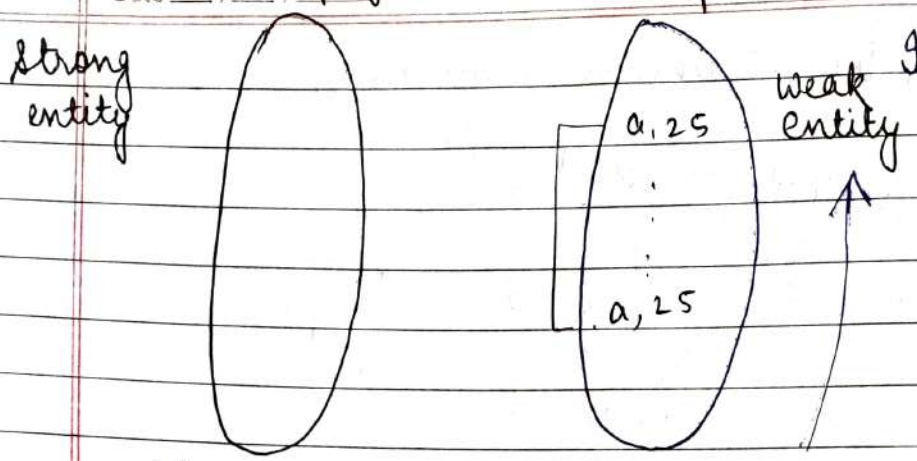



M ⟶ N

Putting the attribute at any side is not feasible.

## DB weak entity

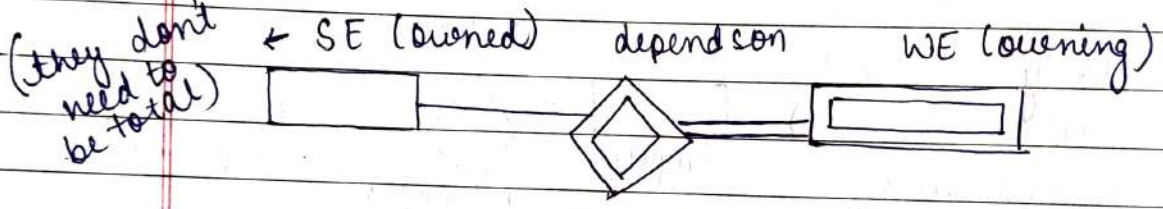
Employee -(Emp Id) dependents




if entity is having primary key

if entity is not having primary key.

Soln. Every weak entity should be related to strong entity.




* The participation should always be total in identifying relⁿ of weak entity.

(id + Name, age)

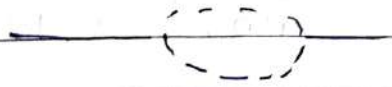
Total participation ⇏ Weak entity

Weak entity ⇒ Total participation

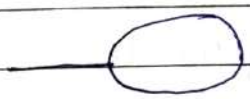
Date / / /

# DB ER diagram notations

Derived attribute

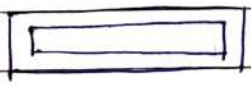


attribute

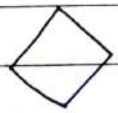


entity

Weak entity

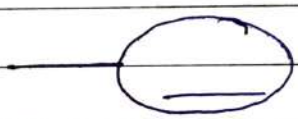


Relationship

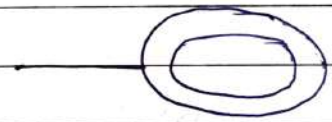


identifying relⁿ.

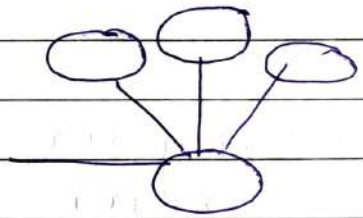
key attribute



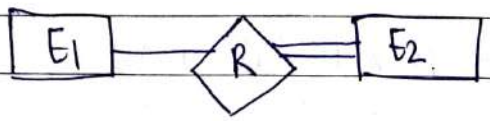
multivalued attribute



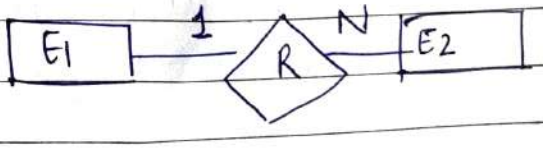
Composite attribute



Total participation of $E_2$ in R




Cardinality ratio
$E_1 : E_2 = 1 : N$



(RDBMS) SQL - SEQUEL - Structured english query language
↓
Simple

Date __/__/__

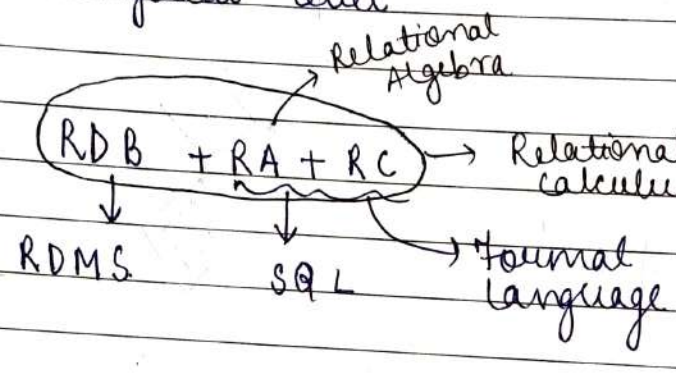
# Relational Database Model

## DB - Introduction to Relational model.

Conceptual level → ER diagrams

(Database model) Representation level → Relational model

Physical level




SQL → RDBMS
↑
Relational → RDB.
DB.

1960s - 1970s (IBM)
legacy Systems (DBMS)
Hierarchal Network (DB)
DB model

Data warehouse → huge data
Databases. → less data

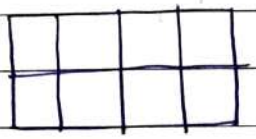
Set theory → relationals → relational DB models.

SQL runs on RDBMS.
Query → converted to relational algebra.

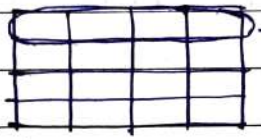
Date __/__/__

# DB - Terminology of Relational database

1) Relation (table)

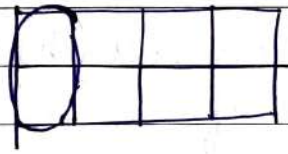


2) Tuple (row) → An entire entity

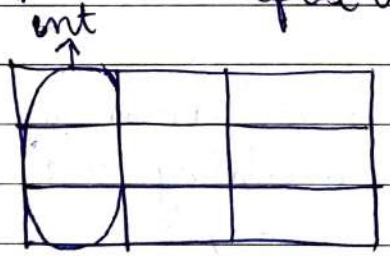



3) Attribute (column)

attribute →



4) Domain Set of names/values eg - set of integers
It is finite . for assumption




5) Relation schema (heading of the table)

attributes $R(A_1, A_2, A_3, A_4, A_5)$

Domains $D_1 \quad D_2 \quad D_3 \quad D_4 \quad D_5$

R

| $A_1$ | $A_2$ | $A_3$ | $A_4$ | $A_5$ |
|---|---|---|---|---|
| | | | | |

$r(R) = D_1 \times D_2 \times D_3 \times D_4 \times D_5$ = an ordered pair containing all values from $D_1, D_2, D_3, D_4, D_5$.

Date __/__/____

The relation is a subset of cartesian product.

$r(R) \subseteq D_1 \times D_2 \times D_3 \times D_4 \times D_5$.

cardinality: $|D_1|$

6) current relation state - The no of elements present in the table at current state.

7) Degree of relation → The no. of attributes in table.

8) intension → relation schema (heading)
9) extension → table itself

DB tuples, tuple values & Null

Relational model - A relation is a set so ordering is something we give.

No 2-tuples would have same value.
No duplicates are allowed.

Exceptions

No value
Eg: Phone no.

Not applicable
Eg: Middle name

Null values

Date __/__/____

# DB constraints on Relational DB schema.

## Domain constraints

In the design level we could put up some restriction on data.
Eg: The value should be etc.

Relational constraints (Restrictions)

- Domain const.
- Key const.
- Entity integrity const.
- Referential integrity constraints

Domain constraints

| Name | | | | → Not allowed |
|---|---|---|---|---|
| SNo | FNo | MNa. | LNa | ✓ Every value should be atomic not divisible. |
| | | | | |
| | | | | |
| | | | | |

Relation - Flat file

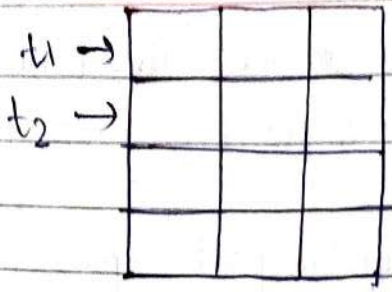
No composite or multivalue attributes are allowed into the domain. You can break it into another table.

Entire schema should be atomic.

## Key constraints

No 2 tuples should have the same values.

Date __/__/____




$t_1 \neq t_2$.

Some attributes can have the same value but not all the tuples.

| SNo | SName | marks. → Super key |
|---|---|---|
| 1 | Ravi | 100 |
| 2 | Sakshi | 50 |
| 3 | Ritu | 40 |

Super key → A subset of attributes which uniquely identifies a table.

Even after deleting SName or marks we can uniquely identify a table.

Key - minimal super key is a key.

If in the worst set of all the attributes in a super key is called a key.

* Any superset of a key is a superkey.

(SNo + SName)

SNo → Key

SName → Super Key

SK → all the attributes

Key → minimize the SK

Key → $A_1$

Attributes → $A_1, A_2, A_3, A_4$

$2 \times 2 \times 2 \times 2 = 8 = 2^4$

minimal - del till no keys are present

Date

How many keys can be there for a relation?
More than one key.

For car. Candidate keys

RN + EN

| ON | color | Price | RN | EN |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |

Candidate keys - If we have more than one minimal superkey for a table then it is called a candidate key.

Primary key - One of the key in candidate key which uniquely identifies a table.

$A_1, A_2, A_3, A_4$ → Superkey (No 2 tuples can have the same value)

$(A_2, A_4, A_3)$ → SK

$(A_3, A_4)$ → SK. and key (minimal superkey)

$A_1\ A_2\ A_3\ A_4$ → SK.

$(A_1\ A_2\ A_4)$ → SK and key

Candidate key = $((A_3 \bullet A_4)\ (A_1\ A_2\ A_4))$.

Primary key (choose Candidate key with less no of attrib)

Date ___/___/_____

# DB Entity and refrential integrity constraint

1) Entity integrity

Refrential integrity

| ENO |
|---|
| |

There should be any null value.

→ No attribute should have a null value.

2) Refrential integrity (Betⁿ 2 or more tables)

Employee FK → Department

| ENO | ENa | Dept No |
|---|---|---|
| 1 | a | 1 |
| 2 | a | X |

| Dept No |
|---|
| 1 |
| 2 |
| 3 |
| 4 |

If the newly joined →

Refrencing

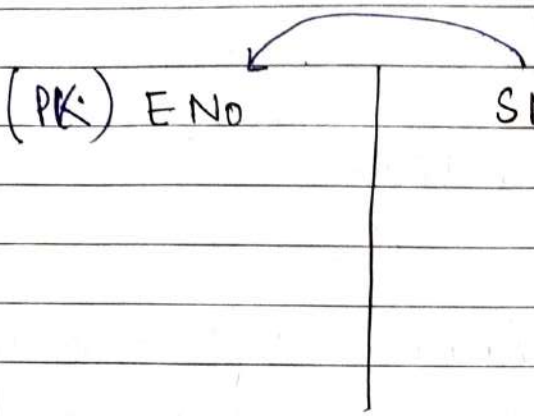
Refrred or base

Foreign key → Primary key on another tabel.

1) Null values are allowed.
2) We can have more than one foreignkey

Date ___/___/_____

In case of recursive relationships referential integrity exists in the same table.




| Primary key | Foreign key |
|---|---|
| 1) A key which uniquely identifies tuples in a table. | 1) Primary key on another table is called foreign key for the given table. |
| 2) Null values not allowed | 2) Null values allowed |
| 3) Only one primary key exists | 3) More than one foreign key can be there |

DB Action upon constraints violations.
When we modify (insertion, updation & deletion) constraints are voilated.

1) Insertion

Date __/__/____

| Ename | Eno | Dep. | Supervisor. |
|---|---|---|---|
| | | | |
| | | | |

Domain constraint → voilated

Key constraint → voilated (duplicate values)
Entity " → null values can be inserted
Referential constraint → value present can't be not some value. voilated

| Can be performed | Domain | Key | Entity | Referential |
|---|---|---|---|---|
| Insertion | X | X | X | X |
| Updation | ✓ | ✓ | ✓ | X |
| Deletion | ✓ | ✓ | ✓ | X 1) (Reject)<br>2) (Cascade) Delete the tuple & the entire tuple from other table<br>3) (Set Null) or some other value |

Date __/__/____

# Counting the number of possible key

1) $R(A_1, A_2, A_3 \cdots A_n)$

$CK = \{A_1\}$. (Smallest super key possible)

↙ Superset of CK is a SK.

| $A_1$ | $A_2$ |
|---|---|
| 1 | a |
| 2 | a |
| 3 | b |
| 4 | b |

$R(\underset{\times}{A_1}, A_2, A_3)$ (2, 2)

$CK = \{A_1\}$

$SK = \underline{A_1}, (\underline{A_1}\ A_2), (\underline{A_1}, A_3), (\underline{A_1}\ A_2 A_3)$

$R(\underline{A_1}, \underbrace{A_2, A_3 \cdots A_n}_{n-2.})$

$SK = 2^{n-1}$

2) $R(A_1, A_2, A_3 \cdots A_n)$

$CK = \{A_1, A_2\}$

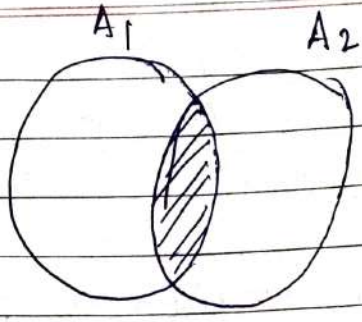
$CK = \{A_1 A_2\}\ 2^{n-2}$

$CK = \{A_1 A_2 A_3\} = 2^{n-3}$.

$CK = \{A_1, A_2\}$

Page No. ____

Date ___/___/_____

CK = 


$SK = SK(A_1) + SK(A_2) - SK(A_1 A_2)$

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

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

3) $R(A_1, A_2, A_3 \cdots A_n)$

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

$SK(A_1) + SK(A_2 A_3) - SK(A_1 A_2 A_3)$

$2^{n-1} + 2^{n-2} - 2^{n-3}$.

$CK = \{A_1, A_1 A_2\}$ → not possible

↓

(SK)

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

$SK = SK(A_1 A_2) + SK(A_3 A_4) - SK(A_1 A_2 A_3 A_4)$

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

Date __/__/____

4) $R(A_1\ A_2 \cdots A_n)$

$CK = \{A_1A_2, A_1A_3\}$. → these is possible because they are not the superset of each other.

$SK(A_1A_2) + SK(A_1A_3) - SK(A_1A_2A_3)$

$2^{n-2} + 2^{n-2} - 2^{n-3}$.

5) $R(A_1\ A_2 \cdots A_n)$

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

$SK(A_1) + SK(A_2) + SK(A_3) - SK(A_1A_2) - SK(A_2A_3) - SK(A_1A_3) + SK(A_1A_2A_3)$

$2^{n-1} + 2^{n-1} + 2^{n-1} - 2^{n-2} - 2^{n-2} - 2^{n-2} + 2^{n-3}$.

6) $R(A\ B\ C\ D)$

$CK(A, BC)$ find SK?

$SK(A) + SK(BC) - S(ABC)$

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

$n=4$

$= 2^{4-1} + 2^{4-2} - 2^{4-3}$

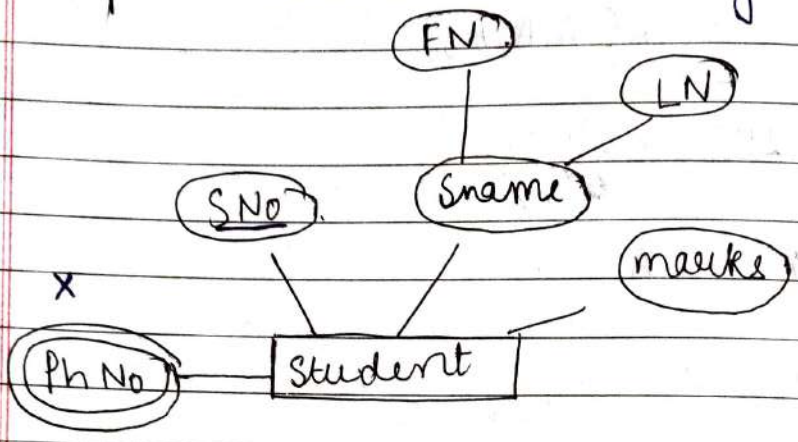
$= 8 + 4 - 2$.

$= 10$ Keys

Date ___/___/___

# Module 3.
# Conversion of ER model to Relational model.

Step 1: Convert the entity into relation



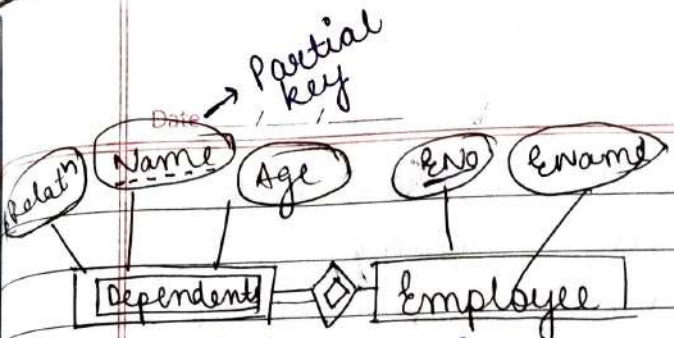

ignore the multivalue attribute in table

Student

| SNo | marks | FN | LN |
|---|---|---|---|
| | | | |
| | | | |
| | | | |

a) Add the simple attribute
b) for composite add the simple attribute in the table
c) Don't add the multivalue attribute

Step 2:




dependent, PK Owner

FK Partial key.

| ENo | Name | Age | Relation |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

primary key.

Owner

| ENo | EName |
|---|---|
| | |
| | |
| | |
| | |

Once you create the relation add all the primary key of owner entity to dependent with full contribution.

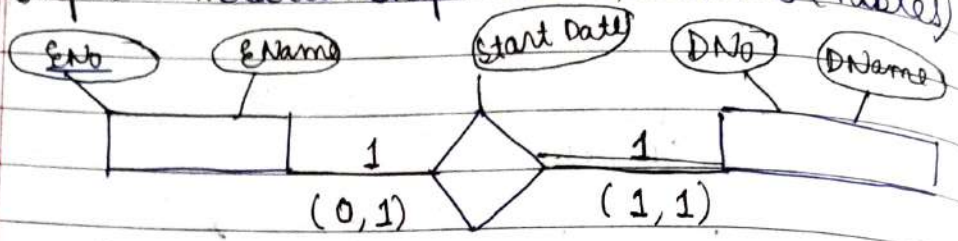
Partial key - You will not identify the tuple uniquely but can use for part of a table.

Need not to take care of the identifying reln.

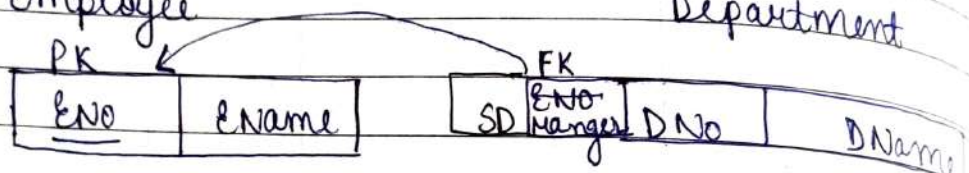
Foreign key → weak entity + identifying.

On del or update on the owner side there should be cascading (i.e. del on both the sides) for weak entity.

Step 3: Relationships to relations (tables)



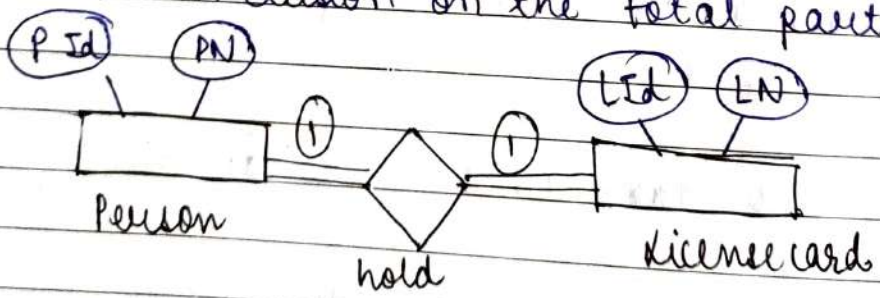

Employee manages Department




Refrenced table Refrencing table

Take the PK on either side or the total partic
side and include it on this side as FK.

* Do the inclusion on the total participating si



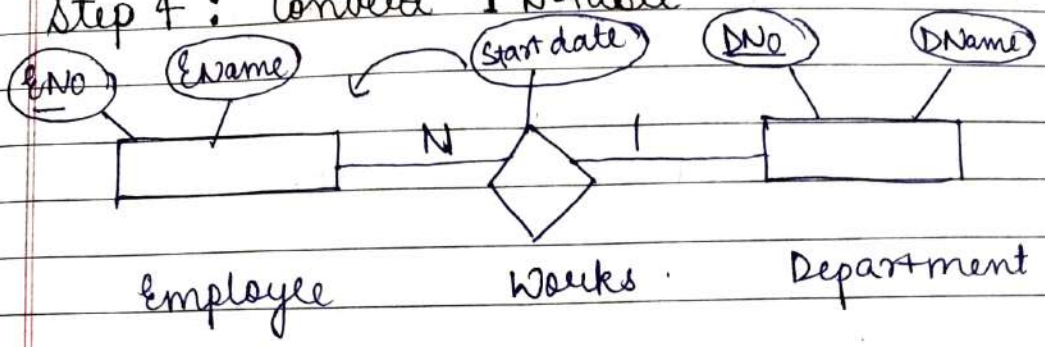

no of entities (person) = no of license card (limit

Combine both the table.

| Person | | license card | |
|---|---|---|---|
| Pid | PN | Lid | LN |
| | | | |

Date ___/___/_____

Step 4 : Convert 1 N-table



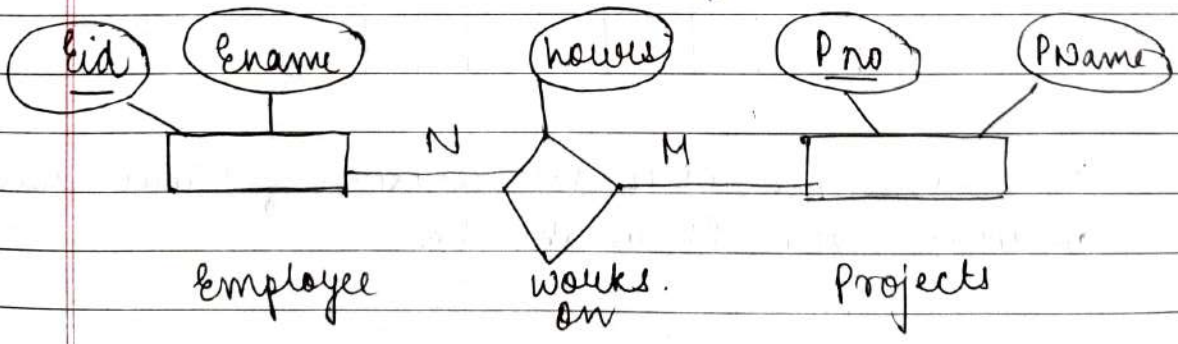

A Dept can have many employees.

Employee

| | PK | | F·K | | Dept PK | |
|---|---|---|---|---|---|---|
| Start date | ENo | EName | DNo | | DNo | DName |
| | | | | | | |
| | | | | | | |
| | | | | | | |

Add the attribute on the one side.

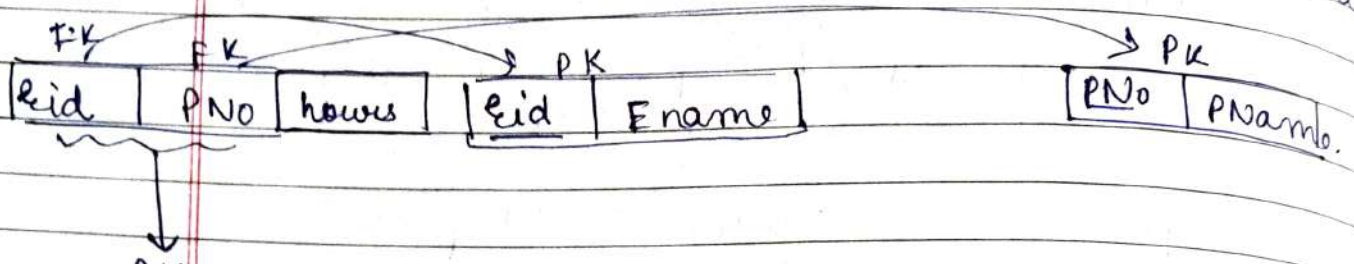
Step 5: Convert Many-Many to table




| Eid | Ename | | Pno | PName |
|---|---|---|---|---|
| | | | | |
| | | | | |

Page No. ____

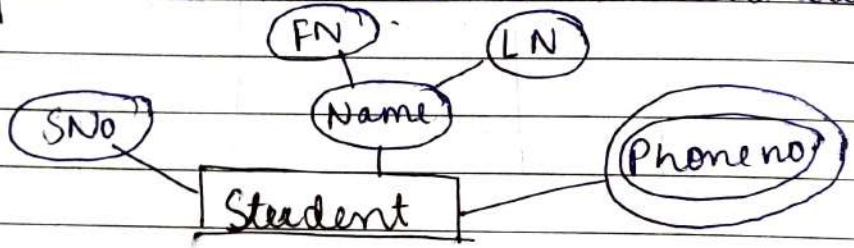
We can't use the foreign key on diff^n tables.

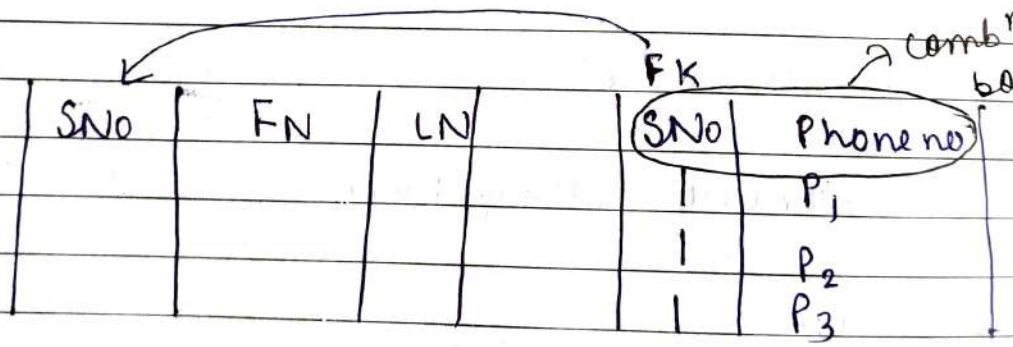



In case of M to N create a new table.

1 - 1

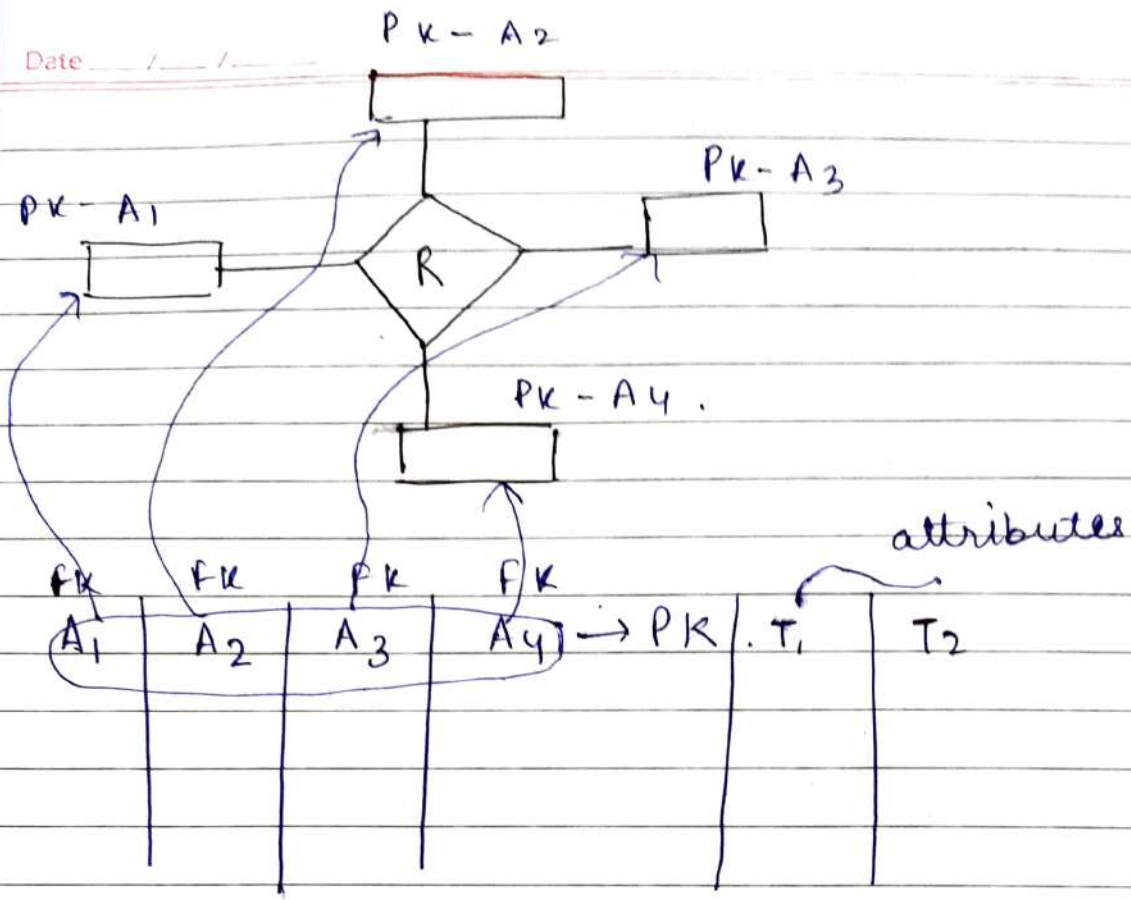
Step 6 : How to deal with multivalued attribute




We create one more new relation for multivalued attribute with PK as the FK.

Step 7 :




## ER model

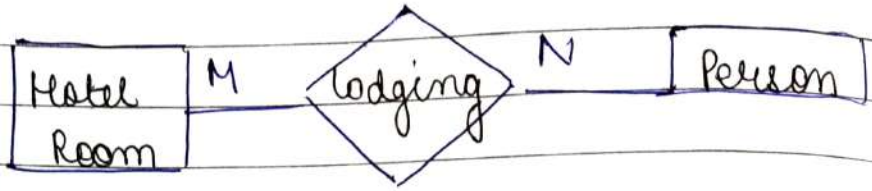
Entity type
1:1 or 1:N
M:N
n ary relationship type
Simple attribute
Composite "
multivalued "
Value set
key attribute

## Relational model

Entity relation
foreign key (or rel$^n$)
relationship (rel$^n$) + 2 FKs.
relationship (rel$^n$) + n FKs.
Attribute
Set of simple component
Relation and foreign key
Domain
Primary key

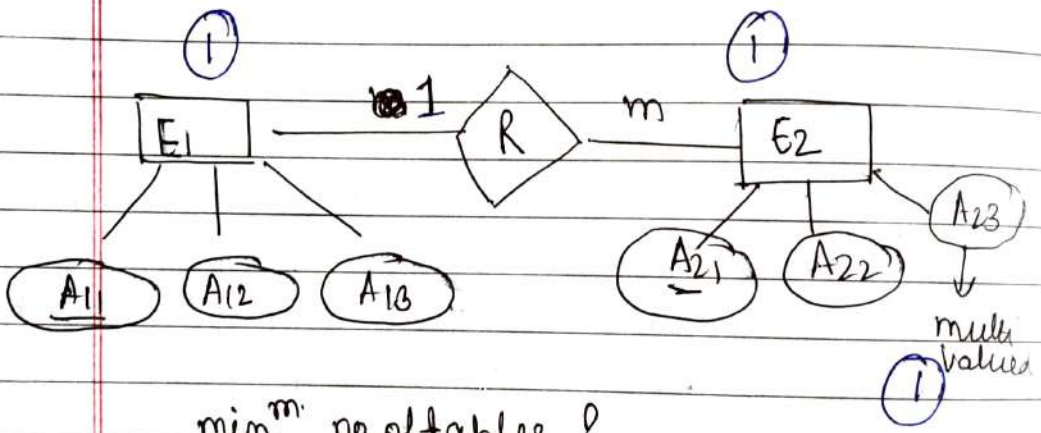
Date __/__/____

# Questions

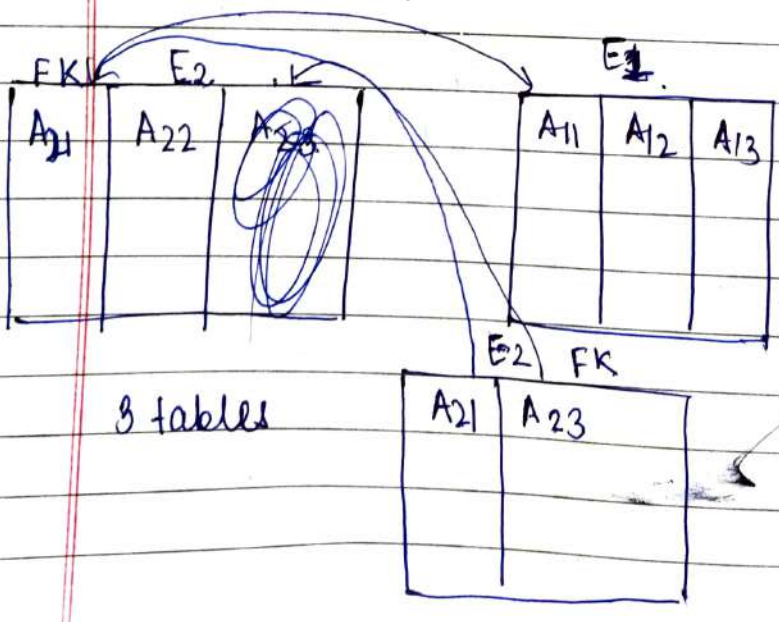



Rent & payment should be added as an attribute to

a) none
b) Hotel
c) Person
d) Lodging




min^m. no of tables ?




3 tables

Eid Sid

Date __/__/__ PK FK.

| A | C |
|---|---|
| 2 x | 4 x |
| 3 | 4 |
| 4 | 3 |
| 5 x | 2 x |
| 7 x | 2 x |
| 9 x | 5 x |
| 6 | 4 |

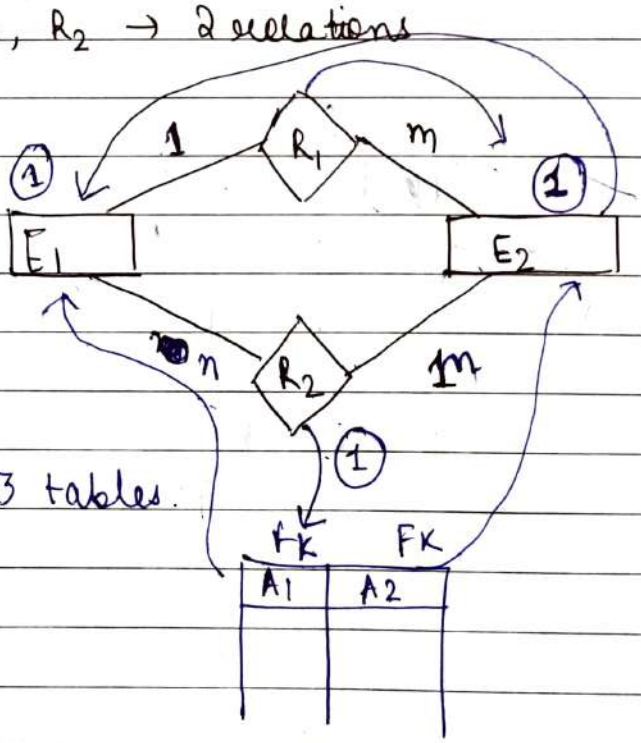
(2, 4)
(2, 5)
(2, 7)
(2, 9)
(5, 9)

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

Which tuples should be deleted when (2, 4) is deleted.

(5, 2) (7, 2) & (9, 5)

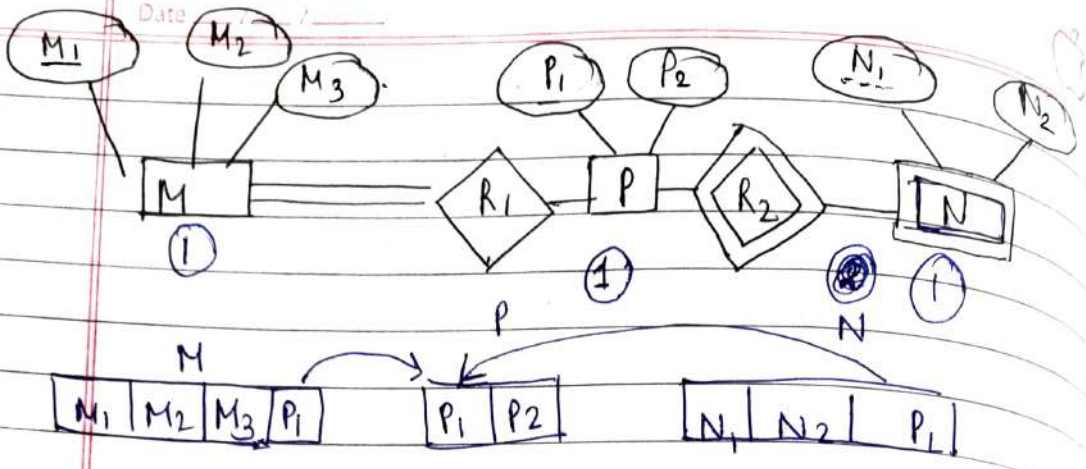
Q. $E_1$, $E_2$ → 2 entities
$R_1$, $R_2$ → 2 relations




3 tables.

$min^m$ no of tables

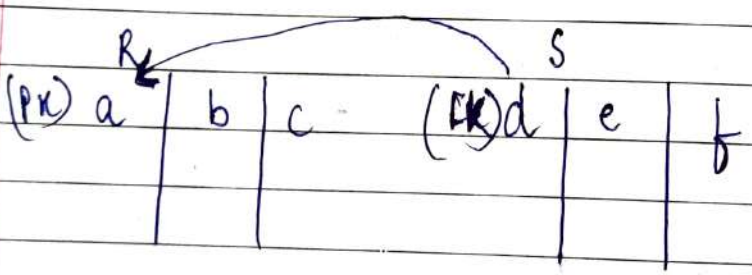
Here we need a table for $E_1$ one for $E_2$
$R_1$ is represented as FK in $E_2$
for $R_2$ we need one table have $PK E_1$ & $PK E_2$.




3 tables

Q. Let R(a, b, c) and S(d, e, f) be 2 rel^n. 'd' is foreign key of S that refers to the primary key of 'R'.

'R' & 'S'




which can cause violation of referential integrity

Insert into S might cause
Delete from R.

Date __/__/____

# Normalisation

↳ Procedure of dividing the table into small tables.

## Introduction

| Eid | EN | Did | Did | Dname |
|---|---|---|---|---|
| (1) | a | 1 | 1 | CS. |

Putting everything we have in one big (universal) table.

1) Redudancy is possible

| Eid | Did | DN |
|---|---|---|
| 1 | 1 | CS |
| 2 | 1 | CS |
| 3. | 1 | CS. |

2) anomlies (problems)
   1) Insert (same information at many places leads to inconsistency)
   2) Deletion
   3) Updation or modification

To the soln. take every table and do spliting of tables called normalisation.

Date __/__/____

# Introduction to Functional dependencies

| A | B | C |
|---|---|---|
| 1 | | |
| (2) | a | b |
| 3 | | |
| 4 | | |
| (2) | a | b |

$A \rightarrow BC$

If we knew values of A then we can find values of B & C.

$$t_1(A) = t_2(A)$$

then

$$t_1(BC) = t_2(BC)$$

| A | B |
|---|---|
| a | 1 |
| a | 1 |
| b | 2 |
| b. | 2. |

$A \rightarrow B$.

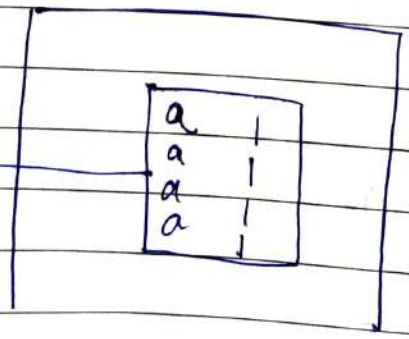
A determines B.

B is functionaly dependent on A

Create a new table

| A | B |
|---|---|
| a | 1 |
| b | 2 |

Areas to be minimized further.



Date __/__/____

$A \rightarrow B$

↓

PK for the table (left)

Normalisation

1NF $\xrightarrow{1}$ 2NF $\xrightarrow{2}$ 3NF $\xrightarrow{3}$ BCNF

$X \rightarrow Y$

X determines Y, Y is functionally dependent on X

$ABC \rightarrow DEF$

| Trivial FD | Non-trivial FD | Semi non trivial FD |
|---|---|---|
| $A \rightarrow A$ | $X \cap Y = \phi$ | $XY \rightarrow YZ$ |
| $AB \rightarrow A$ | $A \rightarrow B$ | $AB \rightarrow BC$ |
| | $A \rightarrow BC$ | |
| $X \geq Y$ | $AB \rightarrow CD$ | Something in common |
| | | $X \cap Y \neq \phi$ |
| Present on the right hand side is present on the left hand side | not present on either side | |

Date ___/___/_____

Rule out the FD based on the tables

| Eid | Ename |
|---|---|
| 1 | a |
| 2 | b |
| 3 | b |

(PK)
Eid → Ename ✓
Ename → Eid ×

| A | B |
|---|---|
| 1 | 1 |
| 1 | 2 |
| 2 | 2 |

A → B ×
B → A ×

A → B for a given value of A B should be unique
1 → 1 ×  1 → 1
1 → 2 ×  1 → 1

| A | B | C |
|---|---|---|
| 1 | 1 | 4 |
| 1 | 2 | 4 |
| 2 | 1 | 3 |
| 2 | 2 | 3 |
| 2 | 4 | 3. |

A → B ×
B → C ×
B → A ×
C → B ×
C → A ✓
A → C ✓

PK

| A | B | C |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 2 | 3 |
| 5. | 3 | 3. |

A → B ✓
BC → A ×
B → C ✓
AC → B ✓

| A | B | C |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 0 |
| 2 | 3 | 2 |
| 2 | 3 | 2 |

A → B ✓
B → C ×

| | | |
|---|---|---|
| 3 | 2 (PK) | 2 |
| x | y | z |
| 1 | 4 | 3 |
| 1 | 5 | 3. |
| 4 | 6 | 3. |

XZ → X ✓  Y → Z
XY → Z ✓  XZ → Y ×
Z → YX

Page No.

Date __/__/____ PK.

| X | y | z. |
|---|---|---|
| 1 | 4 | 2. |
| 1 | 5 | 3. |
| 1 | 6 | 3. |
| 3 | 2 | 2 |

a) $XY \rightarrow Z$ & $Z \rightarrow Y$ X
b) $YZ \rightarrow X$ & $Y \rightarrow Z$ ✓
c) $YZ \rightarrow X$ & $X \rightarrow Z$ X
d) $XZ \rightarrow Y$ & $Y \rightarrow X$ X

R(A, B, C)

| A | B | C |
|---|---|---|
| 1. | 1 | 1 |
| 1. | 1 | 0. |
| 2 | 3 | 2 |
| 2 | 3 | 2 |

1) $A \rightarrow B$ & $B \rightarrow C$ X
2) $A \rightarrow B$ & $B \not\rightarrow C$ X
3) $B \not\rightarrow C$ ✓
4) $A \not\rightarrow B$ and $B \not\rightarrow C$ X

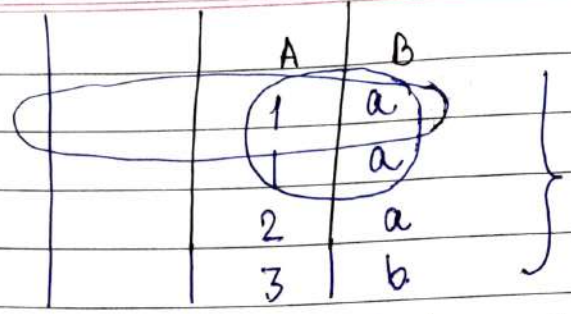
} we can be sure by what doesn't hold but not by what holds From the Req. Analysis only we can define functional dependency

## Formal definition of Functional dependency

| | A | B | $A \rightarrow B$ |
|---|---|---|---|
| $t_1$ | | | |
| $t_2$ | | | |
| | If $t_1$ & $t_2$ agree here | then they must agree here | |
| | If $t_1$ & $t_2$ disagree here | they may agree or disagree | |

Page No.

Date __/__/____




For a given value of A value of B is unique

| A | B |
|---|---|
| 1 | a |

## Various usage of FD

i) identifying the addⁿ. FD
ii) identify the keys
iii) identifying equivalences of FD
iv) Finding minimal FD set

→ two methods

- inference rules
- closure set of attributes

reflexive: $A \to B$ if $B \subseteq A$

transitive: $A \to B$ & $B \to C$ then $A \to C$

Decompositive $A \to BC$ then $A \to B$ and $A \to C$.

Augmentation: $A \to B$ then $AC \to BC$.

Date ___/___/_____

Union

If $A \to B$ and $A \to C$ then

$A \to BC$

Composition.

If $A \to B$ and $C \to D$

then $AC \to BD$

Closure set of rules.

Set of attributes that can be functionally determined from it.

FD: $A \to B$ $B \to D$ $C \to DE$ $CD \to AB$

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

In the table having attributes A, B D

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

$B^+ = \{D, B\}$.

$C^+ = \{D, E, C, A, B\}$ →

$D^+ = \{D\}$

$E^+ = \{E\}$.

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

$(AD)^+ = \{A, D, B\}$

Date ___/___/_____

$A \rightarrow B$

$B \rightarrow D$

$C \rightarrow DE$

$CD \rightarrow AB$.

$AB \rightarrow CD$

$AF \rightarrow D$

$DE \rightarrow F$

$C \rightarrow G_1$

$F \rightarrow E$

$G_1 \rightarrow A$

$(CF)^+ = \{C, F, D, E, G_1, A\}$

$(BG_1)^+ = \{B, G_1, D, A, C\}$

$(AF)^+ = \{A, F, B, E, D, C, G_1\}$. $\{A, F, D, E\}$

$(AB)^+ = \{A, B, D, C, G_1\}$

Determining the candidate key

$R(ABCD)$

$A \rightarrow B$

$B \rightarrow C$

$C \rightarrow D$

$D \rightarrow A$

(A B)

A B AB.

(A B C)

A, B, C, AB, BC, CA, ABC

Page No.

Date

$(A_1, A_2 \cdots A_n)$

Candidate key: $2^n - 1$ candidate key

$R(ABCD) \Rightarrow 2^4 - 1 = 15$

If no candidate key exists then set of all attributes is candidate key.

ABCD. (1)

BCD, ABC, ACD, ABD. (4)

AB, BC, CD, DA, AC, BD (6)

A, B, C, D (4)

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

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

A is a candidate key

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

FDS: $C \to F$ $E \to A$ $EC \to D$ $A \to B$ Key = ?

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

$(CE)^+ = (\widetilde{AB}\ \widetilde{C}\ \widetilde{D}\ \widetilde{E}\ \widetilde{F})$

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

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

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

Page No.

$R = (E, F, G, H, I, J, K, L, M, N)$

FD: $\{ \{E, F\} \rightarrow \{G\}$
$\{F\} \rightarrow \{I, J\}$.

$\{K\} \rightarrow \{M\}$

$\{E, H\} \rightarrow \{K, L\}$

$\{K\} \rightarrow \{M\}$

$\{L\} \rightarrow \{N\} \}$

$EF \rightarrow G$
$F \rightarrow IJ$
$K \rightarrow M$
$EH \rightarrow KL$
$K \rightarrow M$
$L \rightarrow N$.

$(EFH)^+ = (EFGHIJKLMN)$

$2^7$ Super Key.
1 candidate Key

$R = \{A, B, C, D, E, M\}$
FD:

$A \rightarrow B$ $\quad E \rightarrow C$
$BC \rightarrow D$ $\quad D \rightarrow A$

Page No.

Date __/__/____

$(EH)^+ = (ABCDEH)$

d) AEH, BEH, DEH

No of SK =

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

B— $(AEH)^+ = \{A\ E\ H\ C\ B\ D\}$ ✓ (CK)
B— $(BEH)^+ = \{B\ E\ H\ C\ D\ A\}$ ✓ (CK)
$(CEH)^+ = \{C\ E\ H\}$ X
B— $(DEH)^+ = \{D\ E\ H\ A\ B\ D\}$ ✓ (CK)

$SK(K_1 K_2 K_3) = K_1 + K_2 + K_3 - K_1K_2 - K_2K_3 - K_1K_3 + K_1K_2K_3$

Q.

$R = ABCDEFGH$

$F = \{H \to G,\ A \to BC,\ B \to CFH,\ E \to A,\ F \to EG\}$.

$(\_D\_)^+ = (ABCDEFGH)$

X
$D^+ = \{D\}$

✓ $(AD)^+ = \{A, D, B, C, F, H, E, G\}$.
✓ $(BD)^+ = \{B, D, C, F, H, E, G, A\}$. 4 keys.
X $(CD)^+ = \{C, D\}$
✓ $(ED)^+ = \{E, D, A, B, C, F, H, E\}$
✓ $(FD)^+ = \{F, D, E, G, A, B, C, H\}$
X $(GD)^+ = \{G, D\}$
X $(HD)^+ = \{H, D, G\}$

Page No.

Date

not CK  CD , GD , HD.

(CGH)  (GDC)+  (HDC)+
(CGD)  (GDH)+  (HDG)+

(CDG)+
(CDH)+

(CDGH)+ = {C, G, H, D}

4 keys possible.

Q  R = {A B C D E)

AB → C
C → D
B → E

(AB.)+ = ABCDE

CK → (AB)+ = {A, B, E, C, D}.

R = {A, B, C, D)

FD = {AB → CD, C → A, D → B}

A+ = A
B+ = B
C+ = CA
D+ = DB

Page No.

Date / /

✓ $AB^+ = A\ B\ C\ D$

✗ $AC^+ = A\ C$

✓ $AD^+ = A\ D\ B\ C$

✓ $BC^+ = B\ C\ A\ D$

✗ $BD^+ = B\ D$

✓ $CD^+ = C\ D\ B\ A$

B D (A B C D) A C

A C B D

Keys are AB, AD, BC, CD

R = A B C D E F

FD = { AB → C, C → D, D → E, E → F, F → A }

$(B\cdot)^+$ = A B C D E F

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

✓ $AB^+ = \{A, B, C, D, E, F\}$.

✓ $CB^+ = \{C, B, D, E, F, A\}$

✓ $DB^+ = \{D, B, E, F, A, C\}$.

✓ $EB^+ = \{E, B, F, A, C, D\}$

✓ $FB^+ = \{F, B, A, C, D, E\}$

5 keys.

Q. R (A B C D E F)

$A \rightarrow BCDEF$
$BC \rightarrow ADEF$
$DEF \rightarrow ABC$

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

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

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

3 possible candidate keys.

Q20 R = (A B C D E)

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

$A^+ = \{B, C, D, E, A\}$
$xB^+ = \{D, B\}$
$xC^+ = \{C\}$
$xD^+ = \{D\}$
$E^+ = \{A, B, C, D, E\}$

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

$(BC)^+ = \{B, C, D, E, A\}$ $(CD)^+ = \{CDEAB\}$

4 Keys.

Date __/__/____

21 R(ABCD) $AB \to CD$, $D \to A$
what are the CK of sub relⁿ $R_1(BCD)$

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

✗ $BC^+ = \{B, C\}$
✗ $CD^+ = \{C, D, A\}$
✓ $BD^+ = \{D, B, A, C\}$ CK.

$BCD^+ = \{$ Super key $\}$.

22 R(ABCDEF)

$AB \to C$
$B \to D$
$AD \to F$
$C \to D$
$D \to E$
$E \to F$
$E \to D$.

Find CK for $R_1(DEF)$

✓ $D^+ = \{E, D, F\}$
✓ $E^+ = \{E, F, D\}$
✗ $F^+ = \{F\}$.

✗ $EF^+ = \{E, F, D\} \to$ SK
$(DE)^+ \to$ SK
$(DF)^+ \to$ SK

$(DEF)^+ \to$ SK

CK = D, E

Date __/__/____

# Equivalence of FD.

F : { A→C, AC→D, E→AD, E→H }.
G : { A→CD, E→AH }.

F ⊇ G ?  G ⊇ F ?

If every depedency in G is already applied in F

in F(A+) = in G(A+)
‖ ↓
{C, A, D} = {C, D, A}  F ⊇ G

in F(E+) = in G(E+)
↓ ↓
{E, A, H, C, D}  {A, D, E, C, H}.

G ⊇ F

in G(A+) = in F(A+)
{C D} {C, D}
in G(AC+) = in F(AC+)

in G(E+) = in F(E+)
{A, H, D} {AH, D}

Both are equivalent

Page No. ____

Date ___/___/___

# Example on equivalence of FD.

F: { A → B , B → C , C → D }  G: { A → BC , C → D }

## G ⊆ F

| in G | in F |
|---|---|
| $A^+ = \{B, C\}$ | $A^+ = \{B, C, D\}$ |
| $C^+ = \{D\}$ | $C^+ = \{D, C\}$ ✓ |

## G ⊇ F

Each production in G in F

| in F | in G. |
|---|---|
| $A^+ = \{B\}$ | $A^+ = \{B, C\}$ |
| $B^+ = \{C\}$ | $B^+ = \{B\}$ |
| $C^+ = \{D\}$ | $C^+ = \{D\}$. |

F ⊇ G we will see g me vo value hai jo F ko cover kar rahe hai

like F ⊇ G

$A^+ = \{A B C\}$
$C^+ = \{C D\}$.

G ⊇ F

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

30. Check Equivalent FD.

Page No.

Date __/__/____

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

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

For I.

$F \supseteq G_1$

$A^+ = \{ B, A, C \}$ ✓
$D^+ = \{ D, A, C, E, B \}$ ✓.

$G_1 \supseteq F$ X

$A^+ = \{ B, C, A \}$ ✓
$AB^+ = \{ A, B, C \}$ ✓
$D^+ = \{ D, A, B \}$ X no E

not equivalence.

$F \supseteq G_1$
$A^+ = \{ A, B, C \}$ ✓
$B^+ = \{ B, C, A \}$ ✓
$C^+ = \{ C, A, B \}$ ✓

$G_1 \supseteq F$
$A^+ = \{ A, B, C \}$ ✓
$B^+ = \{ B, A, C \}$ ✓
$C^+ = \{ C, A, B \}$ ✓

Equivalent.

Date __/__/____

# DB_ Minimal cover.

Procedure to find Minimal set

1) Split the FDS such that RHS contain single attribute.

$A \to BC \Rightarrow A \to B$ and $A \to C$

2) Find the redudant FDs and delete them from the set

Ex: $\{A \to B, B \to C, A \to C\}$ If closure of $B \to D$ $B^+ = \{D\}$ delete

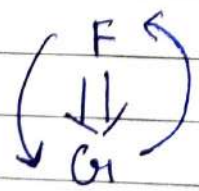
$\Rightarrow \{A \to B, B \to C\}$

3) Find the redudant attributes on LHS and delete them.

Ex: $AB \to C$ $A \to$ can be deleted if $B^+$ contains 'A'.

Minimize

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




minimal - you can't del further

1) $A \to C$
$AC \to D$
$E \to AD$
$E \to H$

$A \to C$
$AC \to D$
$E \to A$
$E \to D$
$E \to H$

Date __/__/____

2) $A^+ = \{A\}$
$AC^+ = \{A, C\}$
$E^+ = \{D, E, H\}$.

$A \to C$
$AC \to D$
$E \to A$
$E \to D.X$
$E \to H$.

$E \to D$ $E^+ = \{A, H, E\}$. ✓
$C, D$

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

2) Delete $E \to D$

$A \to C$ $A^+ = \{A\}$.
$AC \to D$ $AC^+ = \{A, C\}$
$E \to A$ $E^+ = \{D, H, E\}$
$E \to D$ $E^+ = \{A, H, C, D, E\}$ ✓
$E \to H$ $E^+ = \{A, D, C, E\}$.

3) On the LHS

| |
|---|
| $A \to C$ |
| $E \to AD$ |
| $E \to H$ |

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

$AC \to D$

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

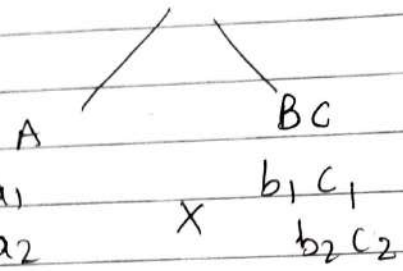
So we will delete 'C'

$A \to C$
$A \to D$
$E \to A$
~~$A \to CD$~~
$E \to H$

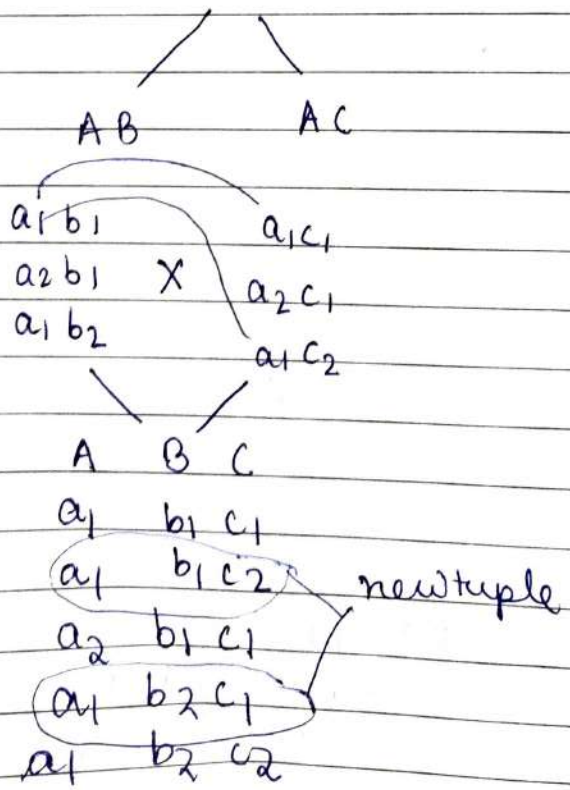
$A \to CD$
$E \to AH$

# Lossless decomposition

| A | B | C |
|---|---|---|
| $a_1$ | $b_1$ | $c_1$ |
| $a_2$ | $b_1$ | $c_1$ |
| $a_1$ | $b_2$ | $c_2$ |




| A | B | C | |
|---|---|---|---|
| $a_1$ | $b_1$ | $c_1$ | |
| $a_1$ | $b_2$ | $c_2$ | |
| $a_2$ | $b_1$ | $c_1$ | |
| $a_2$ | $b_2$ | $c_2$ | new tuple |



Date ___/___/_____

Some times extra information is added is called lossy decomposition

| AB | BC |
|---|---|
| $a_1\ b_1$ | $b_1\ c_1$ |
| $a_2\ b_1$ | $b_2\ c_2$ |
| $a_1\ b_2$ | |

extra tuple → data is lost

Whenever we combine the table we put one or more attribute common to avoid extra data.
So we decompose it along the CK.

| | A | B | C |
|---|---|---|---|
| no extra tuples is added | $a_1$ | $b_1$ | $c_1$ |
| | $a_2$ | $b_1$ | $c_1$ |
| | $a_1$ | $b_2$ | $c_2$ |

If the common attribute is CK then the decomposition is not lossy.

R
/ \
$R_1$ $R_2$

Lossless : while decomposing when a common attribute is CK or key attribute

$(R_1 \cap R_2) \rightarrow (R_1)$ or $R_1 - R_2$
or
$(R_1 \cap R_2) \rightarrow (R_2)$ or $R_2 - R_1$
↓
should be a key

FD preserving

$R(A\ B\ C\ D)$
$R \rightarrow A_1, A_2, A_3 \cdots A_n$
$F \rightarrow F^+$ (set of all FD that can be applied on R)

A→C

$R_1$ (A, B) $R_2$
$F_1 \subseteq F^+$ $F_2 \subseteq F^+$

Page No. ____

Date ___/___/_____

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

i) Original fun. D = later FD
ii) There should be no data loss.

DB decomposition example

R(A, B C) FD : { A → B, B → C, C → A }

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

| $R_1$ | $R_2$ | |
|---|---|---|
| A B | B C | lossless + preserved |
| ✓ A → B | B → C ✓ | |
| ✓ B → A | C → B ✓ | |
| A B → B (trivial, not imp.) | $B^+$ = {C, A, B} ✓ | |
| A B → A (trivial, not imp.) | $C^+$ = {A, B, C} ✓ | |
| A B → A B (trivial, not imp.) | | |
| AB → ϕ | | |

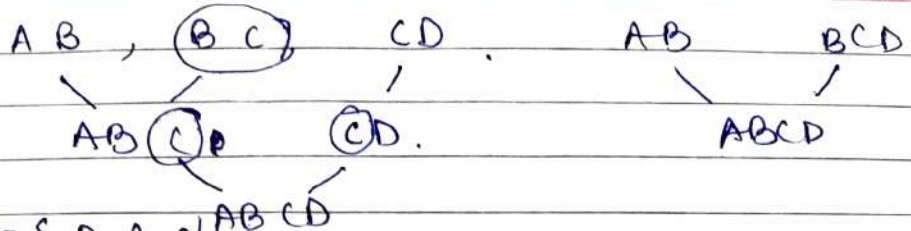
$A^+ = \{A, B, C\}$
$B^+ = \{C, A, B\}$

$R_1 \cup R_2 = R$

Q. R(A B C D)
F = { A → B, B → C, C → D, D → A }
D = (AB, BC, CD)

Page No.

Date ___ / ___ / ___




It is lossless as $C^+ = \{D, A, B\}$

It is lossless

AB ∪ BC ∪ CD.

A → B ✓ B → C $B^+ = (BCDA)$ C → D $C^+ = \{DAB\}$

B → A. ✓ C → B $C^+ = (CDAB)$ D → C. $D^+ = \{DABC\}$

$D^+ = \{D, C, B, A\}$ dependency preserving

lossless - one key in common

preserving - FD tallies or derived are present in set

R(A B C D)

F = { AB → CD , D → A }

D = { AD , BCD }.

| AD | BCD |
|---|---|
| A → D | B → CD |
| D → A | |

AD BCD

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

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

$B^+ = \{B\}$

$C^+ = \{C\}$

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

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

$CD^+ = \{CDA\}$

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

C

AD BCD.

A(D) BC(D)

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

It is lossless

| AD | BCD |
|---|---|
| A → D ✗ | B → B ✗ CD → C |
| D → A ✓ | CD → A |
| | C → C ✗ BD → |
| | D → A ✗ |
| | D → D ✗ |
| | BC → B ✗ |

D → A

It is not preserving

Page No.

Date ___/___/___

Q. R (A B C D E G)

F: A B → C, A C → B, AD → E,
B → D, BC → A, E → G

D: (ABC, ABDE, EG)

1) First see all the attributes are present on the decomposed table or not.

Attribute = A B C D E G ✓ } matching the original table
D = A B C D E G ✓

2) Lossless → find the candidate key.

(AB)C, (AB)DE, EG

If $AB^+$ contains C or DE it's the PK for the table.

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

AB can be the candidate key for one of them

ABC ABDE EG

ABCDE (E)G

Date __/__/____

$E^+ = \{E, G\}$

E can be the candidate key for one of the table.

∴ It is lossless decomposition.

3) Function preserving

Only non trivial dependencies should be taken into account.

| ABC | ABDE | EG |
|---|---|---|
| A→ A X (trivial) | ~~A~~ | |
| B→ D X → D is not in the table (trivial) | B → D | |
| C→ C X (trivial) | E → G X | |
| | AB → DE | |
| 1→ no dependencies found | AD → | |
| $AC^+ = \{ACBDEG\}$ | | |
| $BC^+ = \{BCDAEG\}$ | | |
| $A^+ = \{A\}$ | | |
| $B^+ = \{B, D\}$ | | |
| $C^+ = \{C\}$ | $AD^+ = \{ADEG\}$ | |
| $AB^+ = \{ABCDEG\}$ | | |
| AB → C | | |
| BC → A | | |
| AC → B | | |
| 2→ 3 dependencies | | |

Date __/__/____

# DB INForm. (all the attribute should be atomic)

1) We will go for normalisation in order to remove redudancy

2) In INF attribute values in a table should be atomic, i.e neither multivalued attribute nor composite attributes is allowed.

3) Every table is in INF since the formal definition of a relⁿ guarantee that the attribute values should be atomic (not div

Multivalued

| Eno | Ename | Phone |
|---|---|---|
| 1 | a | 1234 |
| | | 5678 |
| 2 | b | 1212 |
| | | 1342 |

approach1

| ENo | Ename | Phone |
|---|---|---|
| 1 | a | 1234 |
| 1 | a | 5678 |
| 2 | b | 1212 |
| 2 | b | 1342 |

redudent data

approach2

| ENo | Ename |
|---|---|
| 1 | a |
| 2 | b |
| . | |
| | |

| Eno | Phone |
|---|---|
| 1 | 1234 |
| 1 | 5678 |
| 2 | 1212 |
| 2 | 1342 |

Page No.

Date __/__/____

Composite attribute

| ENo | EName | |
|---|---|---|
| | FN1 | FN2 |
| | | |
| | | |
| | | |

=>

| ENo | FN1 | FN2 |
|---|---|---|
| | | |
| | | |
| | | |

nested attributes → composite + multivalued.

To reduce redudancy → normalisation

DB_ Second normal form

1) In 2 NF, we don't allow any partial depedencies.
2) A relation schema R is in 2NF, if every non prime attribute "A" in R is not partially dependent on any key of R.
3) A functional dependency X→Y is a partial dependency, if some attribute "A" ∈ X can be removed from X and the dependency still holds.

| A | B | C |
|---|---|---|
| $a_1$ | $b_1$ | $c_1$ |
| $a_2$ | $b_2$ | $c_2$ |
| $a_1$ | $b_3$ | $c_3$ |
| $a_3$ | $b_3$ | $c_3$ |
| $a_4$ | $b_3$ | $c_3$ |
| $a_5$ | $b_3$ | $c_3$ |
| $a_6$ | $b_3$ | $c_3$ |

$AB \rightarrow C$

$B \rightarrow C$ (partial dependency)

$b_1$ $c_1$

$b_2$ $c_2$

$b_3$ $c_3$.

$AB^+ = ABC$

Key = AB.

NK = C.

AB → ck.

ABC

| AB | BC |
|---|---|
| $a_1$ $b_1$ | $b_1$ $c_1$ |
| $a_2$ $b_2$ | $b_2$ $c_2$ |
| $a_1$ $b_3$ | $b_3$ $c_3$ |
| $a_3$ $b_3$ | |
| $a_4$ $b_3$ | |
| $a_5$ $b_3$ | |
| $a_6$ $b_3$ | |

Page No.

# Third normal form

Even after making the table in 2NF it contains some dependencies or redundancies which need to be removed.

No non key attribute should define other non key attribute.

It is preserving as well as lossless decomp.

Date ___/___/_____

| A | B | C |
|---|---|---|
| 1 | b | x |
| 2 | b | x |
| 3 | b | x |
| 4 | c | y |
| 5 | c | y |
| 6 | c | y |

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

$A^+ = ABC$

K NK
$A \to B$

NK NK
$B \to C$

Transi... depen...

| B | C |
|---|---|
| b | x |
| c | y |

A B C

AB BC

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

$B \to C$.

$AB \cap BC = B$
it is ...
it is ...

Ex1: $R(A\,B\,C\,D\,E)$, $F: \{AB \to C, B \to D, D \to E\}$

Examine whether in 2NF or not

1) Find CK.

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

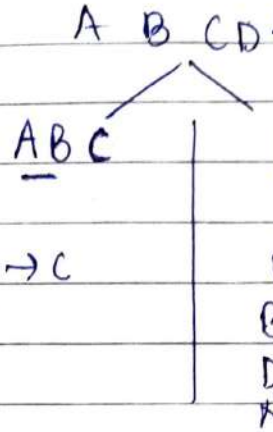
$CK = AB$.

$AB \to C$ $B \to D$ $D \to E$
FD → NK PD → NK : NK NK

$B^+ = \{B, D, E\}$

Page No.

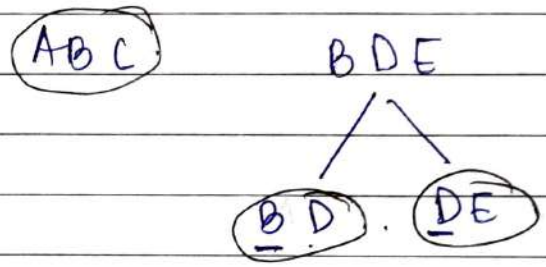
Date __/__/____




It is lossless as well as preserving

$D^+ = \{DE\}$.

For 3rd Normal form




B → D  D → E  no transitive dependency

So, ABC, BD and DE are the tables.

Q. R(ABC) F: {AB → C  C → A}.

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

✓ $AB^+ = \{A, B, C\}$.  $BC^+ = \{B, C, A\}$.

✗ $AC^+ = \{A, C\}$

BC* & AB* is the candidate key

Date __/__/____

BC as well as AB.

$AB \rightarrow C$ $\quad C \rightarrow A$

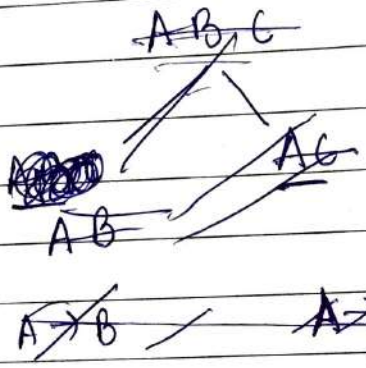
CK PD. PD PD.

not violating 2NF

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

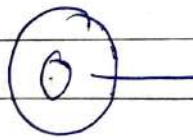
$C^+ = \{C, A\}$

Since all A B C are prime attributes. so it is in 2NF




no NK = NK exists

It is in 3NF




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

$AB \rightarrow C$
$BD \rightarrow EF$
$AD \rightarrow GH$
$A \rightarrow I$
$H \rightarrow J$.

$(ABD)^+ = \{A, B, D, C, E, F, G, H, I, J\}$.

Date __/__/____

ABD is the CK.

AB → C.  BD → EF.  AD → GH
PD NK  PD NK.  PD NK.

A → I  H → J
PD NK  NK NK.

$H^+ = \{ H, J \}$.

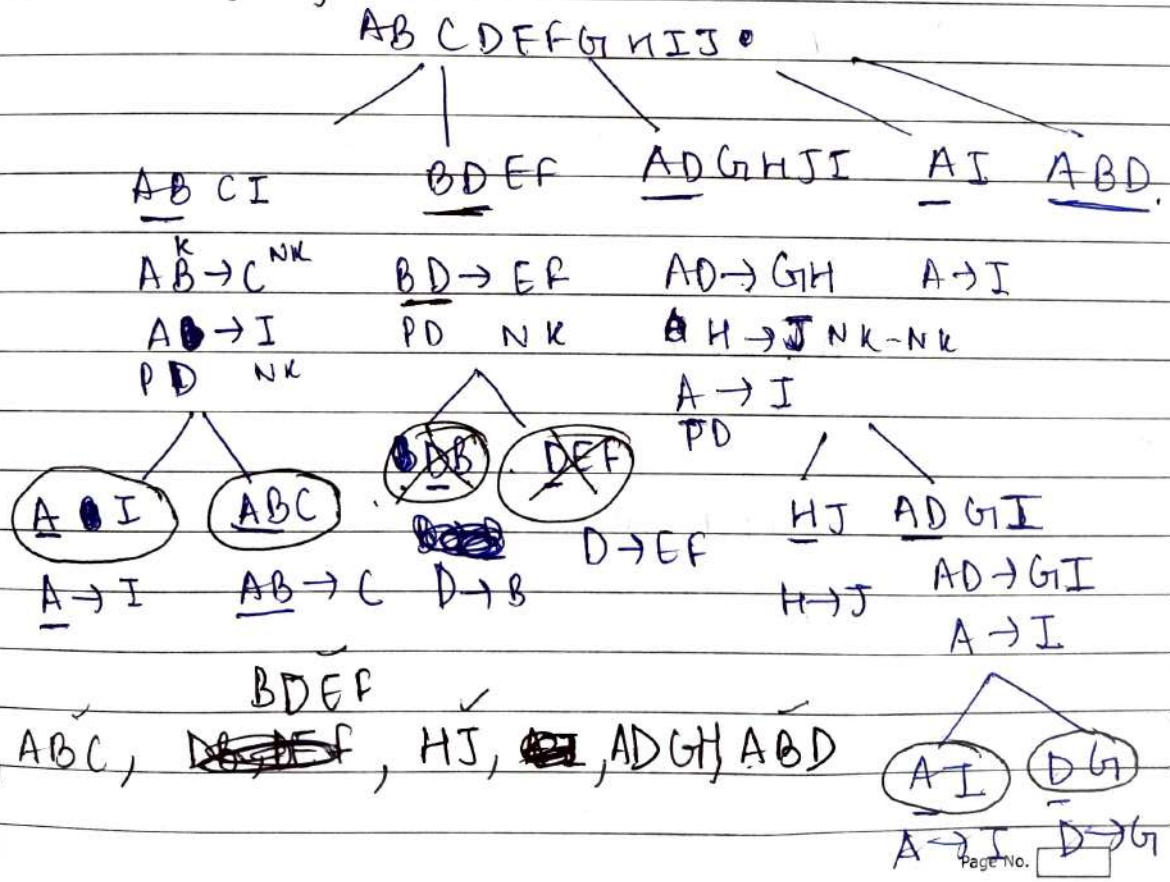
$AB^+ = \{ A, B, C, I \}$

$BD^+ = \{ B, D, E, F \}$.

$AD^+ = \{ A, D, G, H, J, I \}$.

$A^+ = \{ A, I \}$.

$D^+ = \{ D \}$



Date __/__/____

Q. Consider the relation for published book.

Book( $bt^{a_1}$, $An^{a_2}$, $Bty^{a_3}$, $LP^{a_4}$, $Aa^{a_5}$, $P^{a_6}$ )

$bt \rightarrow P, Bty$

$bty \rightarrow LP$.

$An \rightarrow Aa$.

R: ( $A_1$ $A_2$ $A_3$ $A_4$ $A_5$ $A_6$ )

FD: $A_1 \rightarrow A_6 A_3$.

$A_3 \rightarrow A_4$.

$A_2 \rightarrow A_5$.

$(A_1 A_2)^+ = \{A_1, A_2, A_6, A_3, A_4, A_5\}$

CK = $A_1 A_2$.

$A_1 \rightarrow A_6 A_3$. Pd.

$A_3 \rightarrow A_4$ NK.

$A_2 \rightarrow A_5$ Pd

$A_1^+ = \{A_1, A_6, A_3, A_4\}$

$A_2^+ = \{A_2, A_5\}$.

Page No.

Date / /

$A_1 A_2 A_3 A_4 A_5 A_6$.

$\underline{A_1} A_6 A_3 A_4$ $\quad$ $\underline{A_2} A_5$ $\quad$ $\underline{A_1 A_2}$

$A_1 \rightarrow A_6 A_3$. $\quad$ $A_2 \rightarrow A_5$ $\quad$ $A_1 A_2$

$A_3 \rightarrow A_4$. $\quad$ K $\quad$ NK

$A_1 \rightarrow A_6 A_3$ $\quad$ $A_3 \rightarrow A_4$

K $\quad$ NK. $\quad$ NK $\quad$ NK.

P NP ✓

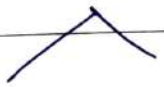
NP P ✓

P P ✓

NP NP

$A_3^+ = \{A_3, A_4\}$.

$A_1 A_6 A_3 A_4$



$\underline{A_1} A_6 A_3$ $\quad$ $\underline{A_3} A_4$.

$A_1 \rightarrow A_6 A_3$. $\quad$ $A_3 \rightarrow A_4$.

$A_1 A_6 A_3$, $A_3 A_4$, $A_2 A_5$, $A_1 A_2$.

## 3NF formal definition (directly in 3NF)

A relational schema 'R' is in 3NF only if in every non trival FD $X \rightarrow Y$ either.

1) X is a superkey (or)
2) Y is a prime attribute

## DB_BCNF introduction

A relational schema 'R' is in BCNF if whenever a non trivial FD $X \to A$ holds in R, then X is a superkey of R i.e. determinants of all FD must be a superkey.

Ex $R(ABC) + \{AB \to C, C \to B\}$

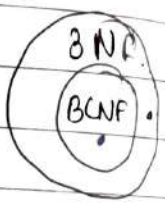
| A | B | C |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 1 | 2 |
| 1 | 0 | 3 |
| 0 | 0 | 4 |
| 4 | 1 | 1 |
| 5 | 1 | 1 |
| 6 | 1 | 1 |
| 7 | 1 | 1 |

C=1
B=1

In 3NF $X \to A$.

1. X is a superkey or
2. A is prime x

All BCNF → 3NF
All 3NF ↛ BCNF




In BCNF

1. X is a superkey

only one condition should be satisfied.

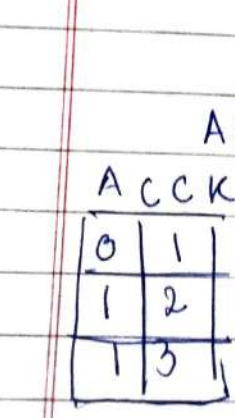
$AB \to C$ SK

$C \to B$ PD

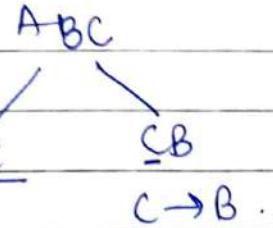
Not in BCNF as C has PD.

$C^+ = \{CB\}$

AC & AB CK



Since FD.




$C \rightarrow B$.

we will never get FD $AB \rightarrow C$ so we will loose the FD. fun. Dependency is not preserved.

BCNF

1) It is lossless
2) It may or may not be FD preserving.

Q. Given R(A B C D E F G H I J) and
F: {$AB \rightarrow C$, $A \rightarrow DE$, $B \rightarrow F$, $F \rightarrow GH$, $D \rightarrow IJ$}
decompose in BCNF.

$AB^+ = \{A, B, C, D, E, F, G, H, I, J\}$.

So, AB is the CK.

| $AB \rightarrow C$ | $A \rightarrow DE$ | $B \rightarrow F$ | $F \rightarrow GH$ | $D \rightarrow IJ$ |
|---|---|---|---|---|
| SK | PD | PD | NP | NP. |

$A^+ = \{A, D, E, IJ\}$ $B^+ = \{B, F, G, H\}$

Date __/__/____


$AB \to C$.

ADE IJ BFGH

$A \to DE$ ✓ $B \to F$

$D \to IJ$ ✗ $F \to GH$.

$D^+ = DIJ$ ADE $F^+ = FGH$ BF

$D \to IJ$ ✓ $A \to DE$ ✓ $F \to GH$.

are in BCNF

DIJ, ADE

Q. R (A B C D E F G H I J) and F: {AB→C, B→D, D→EF, A→GH, H→IJ}

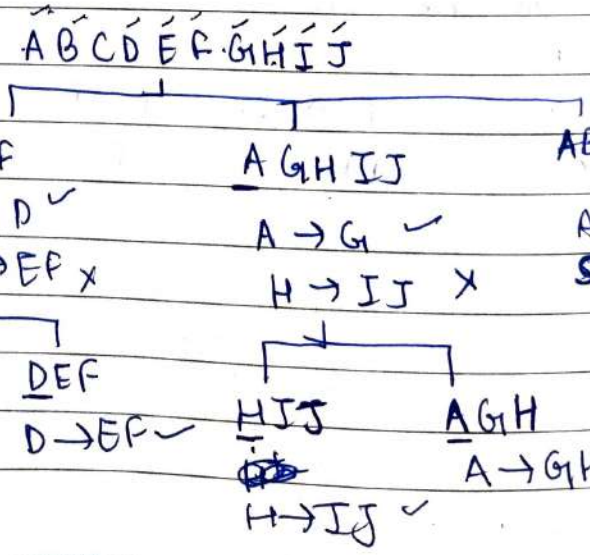
$AB^+ = \{A, B, C, D, E, F, G, H, I, J\}$.

| AB→C | B→D | D→EF | A→GH | H→IJ |
|---|---|---|---|---|
| CK | PD | NP | PD | NP |

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

$H^+ = \{H, I, J\}$

$A^+ = \{A, G, H, I, J\}$




Page No.

Date ___/___/_____

BD, DEF, HIJ, AGH, ABC

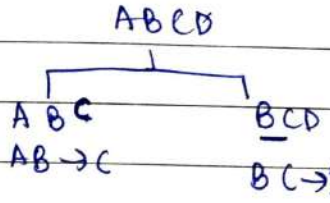
Q. R(A B C D) F : { AB → C , BC → D }.
decompose it to BCNF.

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

AB is the CK.

| AB → C | BC → D |
|---|---|
| CK | NP |

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



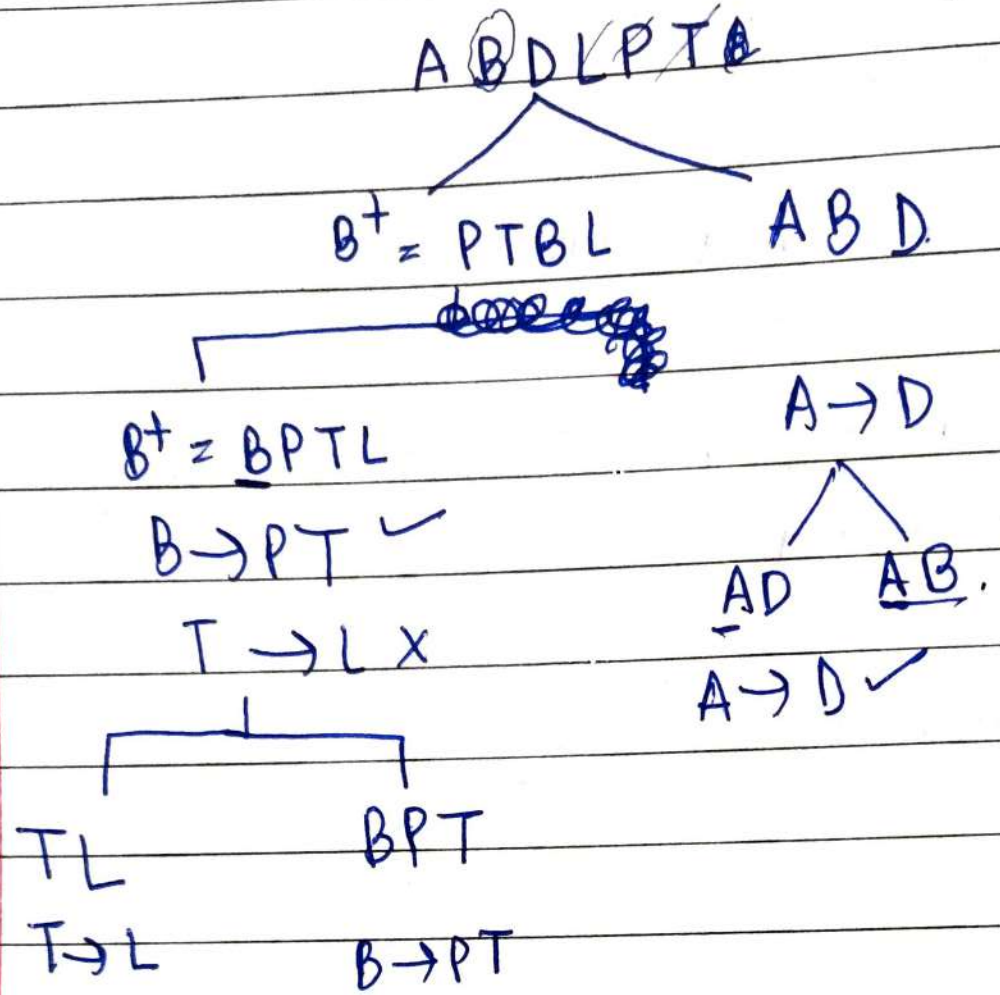

So, tables are ABC and BCD.

Q. R(A B D L P T) and F: { B → PT , T → L , A → D }.
decompose R into BCNF

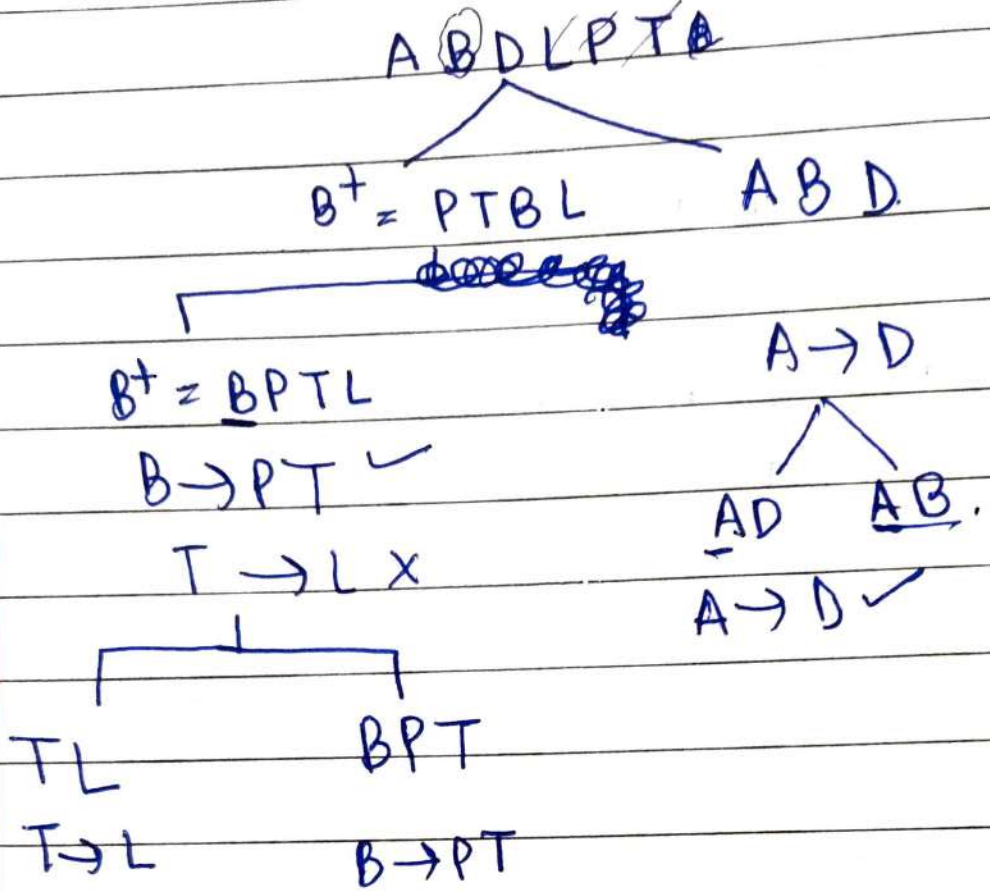
$AB^+ = \{A, B, D, P, T, L\}$

CK = AB

| B → PT | T → L | A → D |
|---|---|---|
| PD | NP | PD |




tables are TL, BPT, AD, AB.

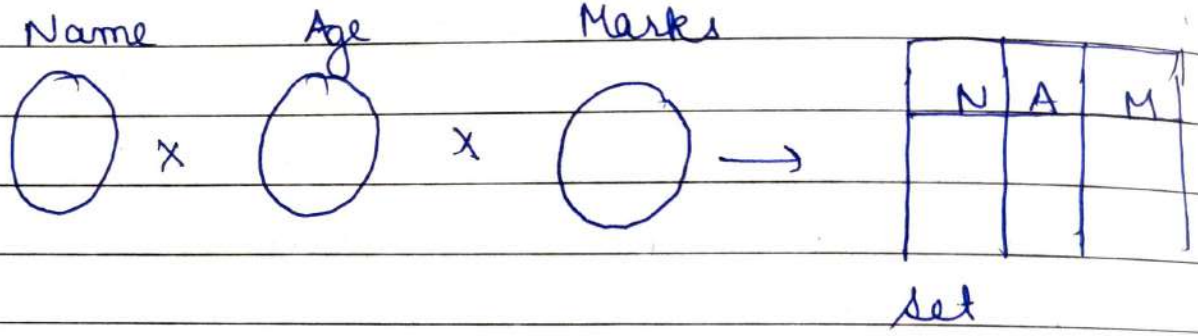



tables are TL, BPT, AD, AB

# Relational Algebra.

implementⁿ RDBMS – SQL

At conceptual level we look at every table as a subset of cross product of all its attributes.




Algo

Relational model: basics for RDBMS

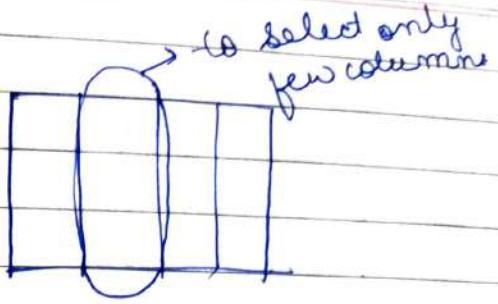
Relational Algebra
what we want & how we want

+ Relational Calculus.
what we want

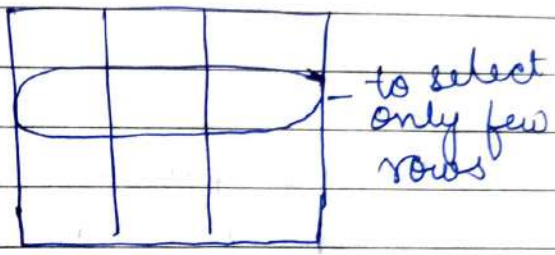
Date __/__/____

Basic Operations (Unary)

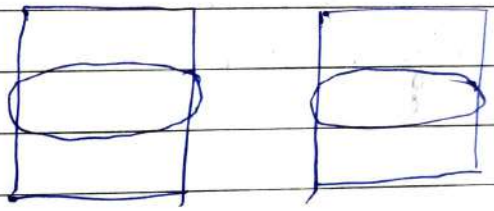
π - projection
(attributes)




σ - selection
(tuples)



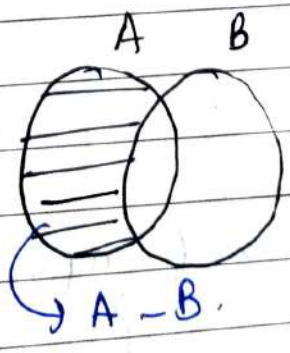

X - cross product
Combining 2 tables of same or diff type



to compare tuples

∪ - union
Combining 2 tables of same type.

- → minus
In A but not in B.




ρ → rename
to rename a table

Date ___/___/_____

Sequence of operⁿ.

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

$\bowtie$ : Join $(\sigma X)$

/ : Division $(\pi, \times, -)$

On cross product every tuple combines with every other table

Join is an extension to cross product for selecting meaningful tuples.

DB_ Selection Operation ($\sigma$) (Horizontal partitioning)

| ENO | Ename | Dno | Salary |
|---|---|---|---|
| 1 | a | 1 | 10K |
| 2 | b | 1 | 11K |
| 3 | c | 2 | 12K |
| 4 | d | 2 | 14K |
| 5 | e | 3 | 15K |
| 6 | f | 3 | 16K |

$\sigma_{DNO=3}$ (Emp)

↑ Relation or relational algebra query whose result is a relⁿ.

Page No.

Date __/__/____

All the employee with Dno = 1 having salaries >10k.

$\sigma_{Dno=1}(\sigma_{Salary>10K}(Emp))$ or $\sigma_{Salary>10}(\sigma_{Dno=1}(Emp))$

Nested expression.

* Selection is commutative
* Degree remains same.

$\sigma_A(\sigma_B(E)) \cong \sigma_B(\sigma_A(E))$

$(\sigma_{Salary>10 \text{ AND } Dno=1}(Emp))$

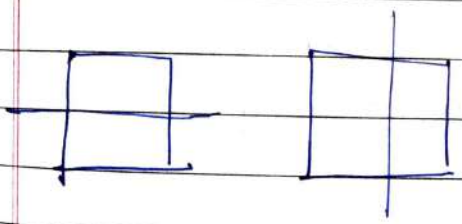
↑ Boolean connectivity

$|\sigma_C(R)|$

min 0 max $|R|$

$0 < \sigma_C(R) < |R|$

DB - projection operation ($\pi$) (vertical partitioning)



$\pi_{x,y}(Emp)$

| X | y |
|---|---|
| 1 | a |
| 1 | b |
| 2 | a |
| 2 | b |

Emp

| X | y | z |
|---|---|---|
| 1 | a | c |
| 1 | b | e |
| 2 | a | c |
| 2 | b | c |

O/p of any RA expression is a relation.

Degree may be variant in the result

Date / /

cardinality is not always same.

$\Pi_Z$ (Emp) = 

| Z |
|---|
| c |

RA is derived from relational model so duplicates are removed.

$\Pi_X$ (Emp)

= 

| X |
|---|
| 1 |
| 2 |

$\Pi_Y$ (Emp) = 

| Y |
|---|
| a |
| b |

Syntax :

$\Pi$ <attribute list> Reln

As long as you project super key on attribute list you might get all the tuples.

* It is not commutative.

$\Pi_X(\Pi_{XY}(Emp)) \neq \Pi_{XY}(\Pi_X(Emp))$

(duplicates allowed) SELECT $\cong$ Projection. (duplicates are not allowed) (in SQL)

SELECT distinct $\cong$ Project.

Date __/__/____

Q. Which of the following query transformations (i.e. replacing LHS by the RHS expression) is incorrect? $R_1$ & $R_2$ are relⁿ's. $C_1$ and $C_2$ are selection conditions and $A_1$ & $A_2$ are attributes of $R_1$.

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

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

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

d) $\pi_{A_1}(\sigma_{C_1}(R_1)) \rightarrow \sigma_{C_1}(\pi_{A_1}(R_1))$

cond$^n$ may be applied on all or any attribute

cond$^n$ is to imposed on $A_1$ only.

b)

| $A_1$ |
|---|
| $a_1$ |
| ~~$b_1$~~ |
| ~~$c_1$~~ |

| $A_1$ | $A_2$ | $A_3$ |
|---|---|---|
| $a_1$ | | |
| ~~$b_1$~~ | | |
| ~~$c_1$~~ | | |

c) $R_1 \cup R_2$

$R_1 \cup R_2$

| $A_1$ | $A_2$ | $A_1'$ | $A_2$ |
|---|---|---|---|
| | | | |
| | | | |
| | | | |

d)

| $A_1$ | $A_2$ | $A_3$ |
|---|---|---|
| | | |
| | | |

| $A_1$ | $A_2$ | $A_3$ |
|---|---|---|
| | | |
| | | |
| | | |

Date / /

Q. Suppose $R_1(A, B)$ and $R_2(C, D)$ are 2 relational schemas. Let $r_1$ & $r_2$ be the corresponding relⁿ instances. B is a foreign key that refers to C in $R_2$. If data in $r_1$ & $r_2$ satisfy referential integrity constraints. Which of the following is always TRUE?

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

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

c) $\pi_B(r_1) = \pi_C(r_2) = \phi$ F

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

$r_1$ $r_2$ (instance name of table)

$R_1$

| A | B |
|---|---|
| | 1 |
| | 2 |
| | |

| C | D |
|---|---|
| 1 | |
| 2 | |
| 3 | |

$\{1, 2\} - \{1, 2, 3\} = \phi$

$\{1, 2, 3\} - \{1, 2\} \neq \phi$

$\{1, 2\} \neq \{1, 2, 3\}$

$\{1, 2\} - \{1, 2, 3\} \neq = \phi$

DB rename operations ($\rho$)

Emp

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

$\pi_{A,B}(\sigma_C(Emp))$

Date __/__/____

| A | B |
|---|---|
| | |

$\rho_{X(C,D)} \pi_{A,B} (\sigma_C (Emp))$

renaming table emp as X and renaming attributes A, B as C, D.

A ⋈ A — join

$\rho_X (Emp)$

Rename employee as X

Only attributes $\rho_{X(C,D,E)} (Emp)$

$\rho_{(C,D,E)} (Emp)$

for few attributes → $\rho_{(X,Y,C)} (Emp)$

$\rho_{X(S,T,C)} (Emp)$

DB_setoperations

$\cup$ $\cap$ - (binary)

RUS → R and S should be union or type compatible.

same domain

R($r_1$, $r_2$, $r_3$) S($s_1$, $s_2$, $s_3$)

Same degree

Page No. ____

Date ___/___/_____

emp_c1 (ename, eid)

emp_c2 (ename1, ei) . . .

union will never contain duplicate values.

R ∪ S = S ∪ R (commutative)

R ∪ (S ∪ T) = (R ∪ S) ∪ T (Associative)

∩

R ∩ S both the tables should be union compatiabile (same attributes & domain)

R ∩ S = S ∩ R (commutative)

R ∩ (S ∩ T) = (R ∩ S) ∩ T (associative)

(R - S) both the tables should be ∪ compatible.

Result would be named as R.

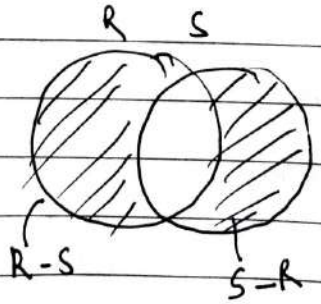
| R | | S | |
|---|---|---|---|
| a | b | c | d |
| 1 | 2 x | 1 | 2 |
| 3 | 4 ✓ | 7 | 8 |
| 5 | 6 ✓ | 9 | 10 |

R - S =

| R | |
|---|---|
| 3 | 4 |
| 5 | 6 |

R - S ≠ S - R (not commutative)

R - (S - T) = (R - S) + T (associative)




R ∩ S = ((R ∪ S) - (R - S)) - (S - R)

R ∩ S = R - (R - S)

# DB- cartesian Product

Cartesian product (X) → we will get ordered pairs.

R (emp)

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

degree = 3 (K)

3

X

S (dep)

| D | E |
|---|---|
| 1 | c |
| 2 | d |

degree = 2 (L)

2.

R X S

degree = K+L

| Empid A | B | C | Empid D | E |
|---|---|---|---|---|
| 1 | a | b | 1 | c |
| 1 | a | b | 2 | d |
| 2 | c | d | 1 | c |
| 2 | c | d | 2 | d |
| 3 | e | f | 1 | c |
| 3 | e | f | 2 | d |

3 X 2 = 6 tuples

|R| = m  |S| = n

|R X S| = |m x n|

To examine we compare 2 tuples from 2 diff^n tables. We can get the name of employee & name of depedent.

$\sigma_{A=D}(R \times S)$

emp  dep

| A | B | C | D | E |
|---|---|---|---|---|
| 1 | a | b | 1 | c |
| 2 | c | d | 2 | d |

In general we use cartesian product with selection to get some meaningful data.

Date ___/___/___

DB_Join ($\bowtie$) → subset of cartesian product

(extension for cartesian product)

X, a $\bowtie$

$\sigma_{A=D}(R \times S) \cong R \bowtie_{<A=D>} S$

(R) (S)

$R \bowtie_{<join\ cond^n>} S$

$cond^n 1$ AND $cond^n 2$

A=D AND B=E

$R \bowtie_{<A=D \text{ AND } B=E>} S$ (k+l) attributes

R

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

S

| D | E |
|---|---|
| | |

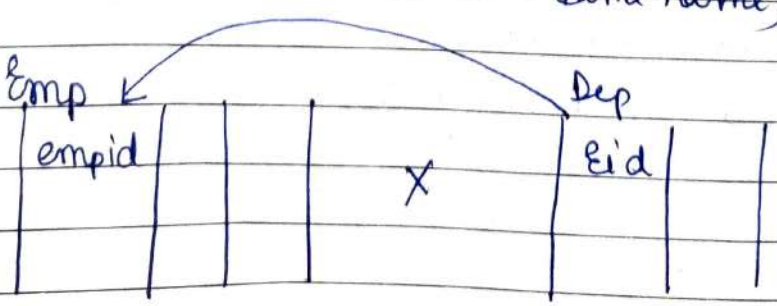
θjoin = only comparing 2 attributes (A=10) X not join

$0 < R \bowtie S < m \times n$

DB_Natural join (*)

↓

(2 tables, common attribute same name)



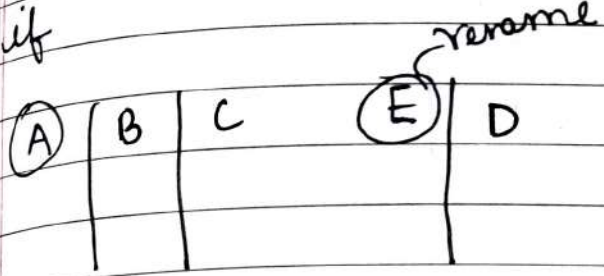
Date __/__/____

$R * S \cong R \bowtie_{<A=A>} S \cong \sigma_{<A=A>}(R \times S)$

| A | B | C | A | D |
|---|---|---|---|---|

(should have the same)

if




$R * \rho_{(A,D)}(S)$

| A | B | C | A | D |
|---|---|---|---|---|
| 1 | a | b | 1 | d |
| 2 | c | d | 3 | e |

| A | B | C | D |
|---|---|---|---|
| 1 | a | b | d |

Common attributes needs not be written 2 times

| A | B | C | A | B |
|---|---|---|---|---|
| 1 | a | b | 1 | a |
| 2 | c | d | 3 | e |

| A | B | C |
|---|---|---|
| 1 | a | b |

1) attributes might change $0 < R * S < m * n$
2) can be applied on any no. of tables

Page No. ____

Date __/__/____

When names of attributes are not same then natural join is equal to the cartesian product

$A_1$

| A | B | C |
|---|---|---|
| 1 | a | b |
| 2 | c | d |

$A_2$

| D | E |
|---|---|
| 1 | a |
| 3 | e. |

$A_1 \bowtie A_2$

| A | B | C | D | E |
|---|---|---|---|---|
| 1 | a | b | 1 | a |
| 1 | a | b | 1 | a |
| 2 | c | d | 3 | e |
| 2 | c | d | 3 | e |

## DB- Division operation ($R \div S$)

Implementation of Division with basic operation

1. $T_1 \leftarrow \Pi_{(R-S)}(R)$
2. $T_2 \leftarrow \Pi_{(R-S)}((S \times T_1) - R)$
3. $T \longleftarrow T_1 - T_2$

$R \div S$

R

| A | B |
|---|---|
| $a_1$ | $b_1$ |
| $a_2$ | $b_1$ |

S

| A | B |
|---|---|
| | |

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

Page No.

Date ___/___/___

R

| A | B |
|---|---|
| $a_1$ | $b_1$ |
| $a_2$ | $b_1$ |
| $a_1$ | $b_2$ |
| $a_2$ | $b_2$ |
| $a_1$ | $b_3$ |

S

| A |
|---|
| $a_1$ |
| $a_2$ |

| B |
|---|
| $b_1$ |
| $b_2$ |

$b_1$ & $b_2$ is present in combⁿ with both $a_1$ & $a_2$

1) $T_1 \leftarrow \pi_{(R-S)}(R)$ $\quad \pi_B(R) =$ $T_1$

| B |
|---|
| $b_1$ |
| $b_2$ |
| $b_3$ |

2) $T_2 \leftarrow \pi_{(R-S)}((S \times T_1) - R)$

$S \times T_1 =$

| A | B |
|---|---|
| $a_1$ | $b_1$ |
| $a_1$ | $b_2$ |
| $a_1$ | $b_3$ |
| $a_2$ | $b_1$ |
| $a_2$ | $b_2$ |
| $a_2$ | $b_3$ |

$(S \times T_1) - R$

| A | B |
|---|---|
| $a_1$ | $b_1$ |
| ~~$a_1$~~ | ~~$b_2$~~ |
| ~~$a_1$~~ | ~~$b_3$~~ |
| $a_2$ | $b_1$ |
| ~~$a_2$~~ | ~~$b_2$~~ |
| $a_2$ | $b_3$ |

$T_2 =$

| B |
|---|
| $b_3$ |

Page No. ____

Date __/__/____

3) $T \leftarrow T_1 - T_2$

| B |
|---|
| $b_1$ |
| $b_2$ |

## DB- Complete set of RA operations

We can say that the following set of RA operations $\{\sigma, \pi, \cup, -, \times\}$ is a complete set that is any of the other RA operⁿ can be expressed as a sequence of operⁿ from this set.

$\cap$ → $-$, $\cup$

$\bowtie$ → $\sigma$, $\times$

$-$ → $\times$, $\sigma$, $-$

## DB- Types of Join

Left ~~outer~~ outer join

Right outer join

Full outer join

R

| A | B |
|---|---|
| 1 | a |
| 2 | c |

S

| C | D |
|---|---|
| 1 | b |
| 3 | d |

A=D

| A | B | C | D |
|---|---|---|---|
| 1 | a | 1 | b |
| Null | Null | 3 | d |

Date __/__/____

$R \bowtie_{A=C} S$

| A | B | C | D |
|---|---|---|---|
| 1 | a | 1 | b |

$R \bowtie_{A=C} S$

| A | B | C | D |
|---|---|---|---|
| 1 | a | 1 | b |
| 2 | c | Null | Null |

Left outer join: $R \bowtie_{A=C} S$.

whatever present on the left side & not included in the table should be present there.

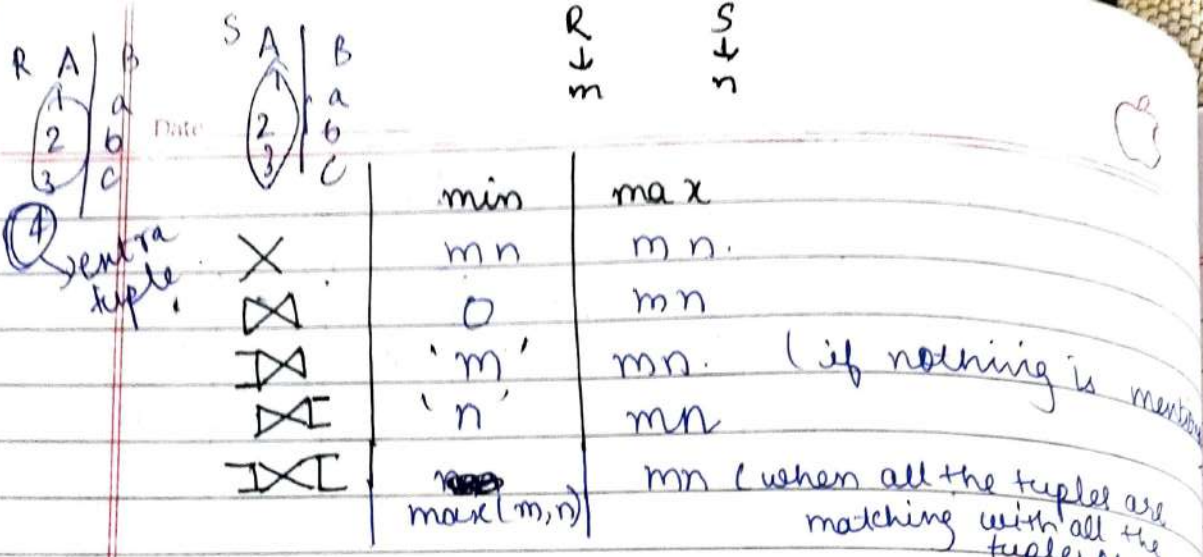
Right outer join: $R \bowtie S$.

whatever present on the right side & not selected, should be present in the o/p.

$R \bowtie S$ (present on both the sides)

| A | B | C | D |
|---|---|---|---|
| 1 | a | 1 | b |
| 2 | c | N | N |
| N | N | 3 | d |

$\times \quad \bowtie \quad \bowtie \quad \bowtie \quad \bowtie$




## 14. DB - Extended RA operations

**Generalized projection** : Can use arithmetic operations such as +, −, *, ÷ on numeric attributes, numeric constants and on an expression that generate numeric result. Also permits operations on other datatypes such as concatenation of strings.

eg: $\Pi_{ID, name, dept\_name, (salary \div 12)}$ (Instructor)

**Aggregate function** : (SUM, AVERAGE, MAX, MIN, COUNT, COUNT-DISTINCT)

$\mathcal{F}_{\langle functionlist \rangle}(R)$ $\mathcal{F}_{COUNT\ SSN,\ AVERAGE\ SALARY}$ (Employee)

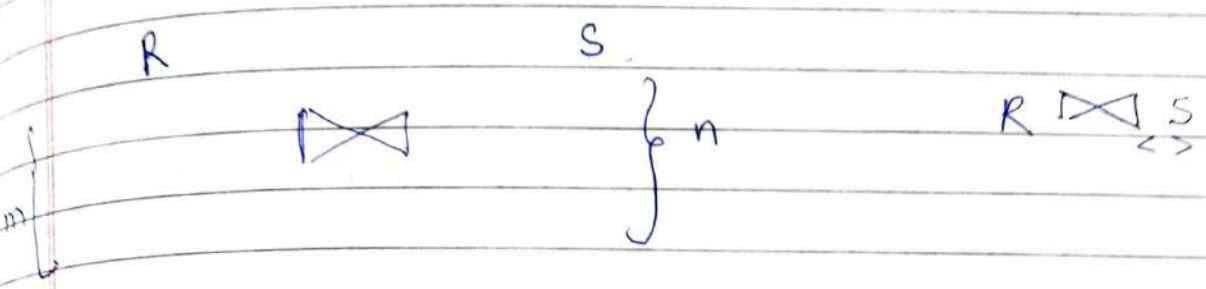
SSN → Social security number.

Aggregate functions with GROUP BY.

$\rho_R$ (DNO, No.of.Emp, Avg_Sal) ( $\mathcal{F}_{DNO\ COUNT\ SSN,\ AVERAGE\ SALARY}$ (Emp) )

2. Consider the join of a relation R with a relation S. If R has m tuples and S has n tuples, then the $max^m$ and $min^m$ sizes of joins are respectively?

a) $m+n, 0$ b) $mn, 0$ c) $m+n \ |m-n|$
d) $mn, m+n$

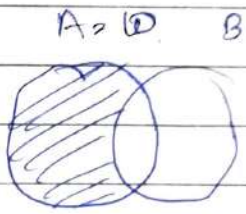



$min^m = mn, 0$

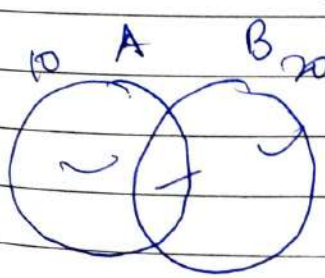
3. The relational algebra expression equivalent to the following tuple condition calculus expression

$$\{t \mid t \in r \wedge (t[A] = 10 \wedge t[B] = 20)\}$$

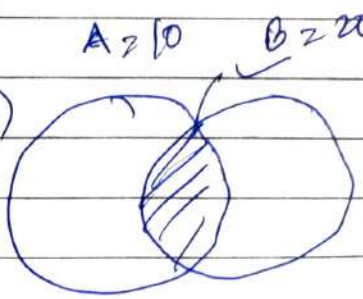
a) $\sigma_{(A=10 \vee B=20)}(y)$
b) $\sigma_{A=10}(r) \cup \sigma_{(B=20)}(y)$
c) $\sigma_{A=10}(r) \cap \sigma_{(B=20)}(y)$
d) $\sigma_{A=10}(r) - \sigma_{(B=20)}(y)$




d)




b)




c)

Date __/__/____

# Introduction to relational Calculus.

Formal lang.

(Procedural) Relational Algebra
i) what data you want from rel^n.
ii) how you can get it (relations)

i) What we want (declarative lang)

Relat^n Calculus

- Tuple RC
- Domain RC

They will helpful to implement any query pr.

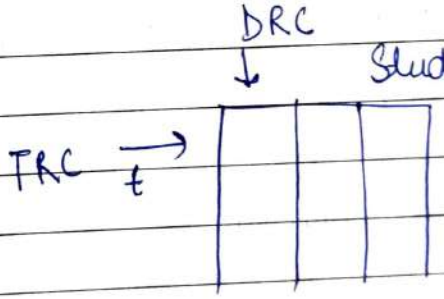
* If a language is able to express everything a RA can then it is called as relationally complete.

TRC, DRC, RA → relationally complete

↑ we generally range over the columns.

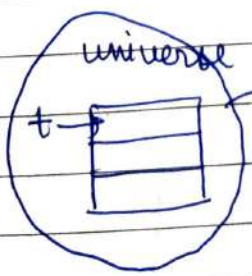
↓

take each tuple at a time & examine




Unsafe operat^n. Student (t)

~t → all the tuples not in student table. (infinite).




RA and RC are of same power but due to unsafe operat^n. RA is more powerful than RC (Relational calculus)

Date ___ / ___ / ___

# DB TRC Syntax

Student

| FN | LN | marks |
|---|---|---|
| a | b | 100 |
| b | c | 40 |
| c | f | 60 |

FN of student having marks > 50

$$\{ t.FN \mid student(t) \text{ AND } t.M > 50 \}$$

$t.FN$ — set of all attributes in final o/p.

$student(t)$ — tables AND

$t.M > 50$ — condition

$$\{ t_1.A_1, t_2.A_2 \ldots \mid T_1(t_1) \text{ AND } T_2(t_2) \ldots \text{ AND } t_1.A_1 \}$$

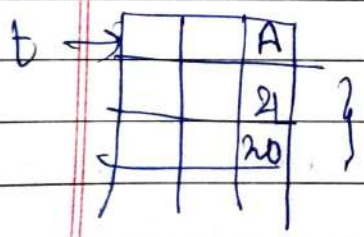
## DB – FREE AND BOUNDED variables.

$$\{ t_1.A_1, t_2.A_2 \mid \}$$

Bounded variables

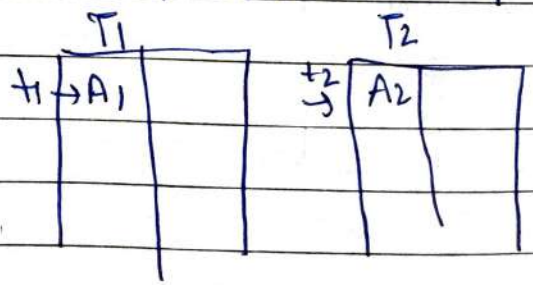
expressions (atomic (simple) or composite (complex))

Emp




→ Emp(t)
→ t.A > 20
→ $t_1.A_1 > t_2.A_2$

$A_1 \wedge A_2$ AND
$A_1 \vee A_2$ OR
$\neg A_1$ NOT
$\exists t(\,)$ Exist
$\forall t(\,)$ for all



Date __/__/____

Unit for processing → Entire table | One tuple.

Bounded variables | Free variable

$\exists t$ ( ) — there exists t — having quantifiers | no quantifiers

$\forall t$ ( ) — for all t — having quantifiers

$\exists t\,(t(A) > 50)$ false

there exists atleast one tuple such that a A value is 50.

| A | B |
|---|---|
| 10 | 1 |
| 20 | 2 |
| 30 | 3 |
| 50 | 4 |

$\forall t\,(t(B) < 10)$

True

processing the entire table | processing step by step.

In bounded variable we have to check the entire table i.e if $t(B) < 10$ then only it will return true or false but in case of free variable only one tuple needs to checked each time.

$\sim \exists t\,(t(A) > 50)$ ↓ doesn't exist ; $\exists t\,(t(A) > 50)$ ↓ there exists

Page No. ____

Date / /

Q19. Let r be a relation instance with schema $R = (A, B, C, D)$. We define $r_1 = \pi_{A,B,C}(r)$ and $r_2 = \pi_{A,D}(r)$. Let $S = r_1 * r_2$, where * denotes natural join. Given that the decomposition of r into $r_1$ & $r_2$ is lossy. Which one is true?

i) $S \subset r$

ii) $r \cup S = r$.

iii) $r \subset S$

d) $r * S = S$.

$r_1$

| A | B | C |
|---|---|---|
| 1 | $b_1$ | $c_1$ |
| 1 | $b_2$ | $c_2$ |
| 2 | $b_3$ | $c_3$ |
| 2 | $b_4$ | $c_4$ |

$r_2$

| A | D |
|---|---|
| 1 | $d_1$ |
| 1 | $d_2$ |
| 2 | $d_3$ |
| 2 | $d_4$. |

r

| A | B | C | D |
|---|---|---|---|
| 1 | $b_1$ | $c_1$ | $d_1$ |
| 1 | $b_2$ | $c_2$ | $d_2$ |
| 2 | $b_3$ | $c_3$ | $d_3$ |
| 2 | $b_4$ | $c_4$ | $d_4$ |

$S = r_1 * r_2$

| A | B | C | D |
|---|---|---|---|
| | | | |

R (ABCD)

$r_1$ ABC  AD $r_2$

*

ABCD S

lossy - after joining we get more tuples.

$r \subset S$

final table must be a superset of initial table

| | | | |
|---|---|---|---|
| 1 | $b_1$ | $c_1$ | $d_1$ |
| 1 | $b_1$ | $c_1$ | $d_2$ |
| 1 | $b_2$ | $c_2$ | $d_1$ |
| 1 | $b_2$ | $c_2$ | $d_2$ |

S.

Page No.

Date __/__/____

# Introduction to DRC

SQL - most prac. lang.

RA + TRC = SQL

DRC ≈ QBE

TRC

R

| A | B | C |
|---|---|---|
| → | | |

t.A  t.B

t.C

DRC

R  a  b  c

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

no of variables is more.

{ _____ | _____ }.

what to project

t.a = 10 → TRC

{ | R(a, b, c) ∧ a = 10 } — DRC

free

bounded

OR

{ | ⟨a, b, c⟩ ∈ R ∧ a = 10 }.

# SQL

E.F Codd - A relational model of data for large shared data banks.

RAYMOND and DONALD: SEQUEL

Structured English query language

SQL

1) It is based on RA and TRC
2) First implement on system R by IBM
3) Oracle, Microsoft SQL server, MS access, MySQL etc
4) Portable
5) Basics and extensions

## Creation of Schema

Schema: Groups logical objects together.

For privacy concerns we group objects together.


Syntax: CREATE SCHEMA Schema_name [AUTHORIZATIO
owner_nam

CREATE SCHEMA FINANCE AUTHORIZATION RAVI;
GRANT SELECT ON FINANCE *. to user1;

Date __/__/____

## Create Table

Syntax: CREATE TABLE <tablename> (<col1> datatype (width),
<col2> datatype (width); ...);

Eg: CREATE TABLE Customer( Name varchar2(20),
cust_id Number,
Address varchar2(50));

Datatypes: We have numerous datatypes in SQL

| Numeric | Character | Date | Boolean |
|---|---|---|---|
| NUMBER | CHAR | DATE | BOOLEAN |
| FLOAT | VARCHAR2 | TIME | |
| INT | NCHAR | TIMESTAMP | |
| REAL | NVARCHAR2 | INTERVAL | |
| DECIMAL | LONG | | |
| BINARY | RAW | | |
| FLOAT | LONG RAW | | |

Constraints: We have 7 constraints in SQL

1) NOTNULL (the attribute shouldn't contain null value)
2) UNIQUE (no two tuples can have same value)
3) Primary key ( should be unique + NOTNULL)
4) Foreign key ( referential integrity (attribute in table referring to attribute of other table)
5) CHECK ( sometimes we need to check any attribute)
6) DEFAULT ( default value is set when there is no value)
7) REF constraints (

Date ___ / ___ / ____

Our requirement is:
Dept table with dep_id(PK) dept_name (UNIQUE) location.

```
CREATE TABLE DEPARTMENT(
  deptid NUMBER PRIMARYKEY,
  dept_name varchar2(30),
  location varchar2(100),
  UNIQUE (dept_name) );
```

Constraints could be defined at 2 levels.

a) column level ↓

dept_id NUMBER PRIMARYKEY

Specifying the constraints along with the column name.

b) table level

dept_name varchar2(30)
UNIQUE (dept_name));
Specifying the constraint at the last of the columns.

* NOT NULL should be specified at column level
* composite key should be specified at table level. (more than one attribute as Key)

Emp: emp_id (PK) emp_name (NOT NULL)
job, dept_Id (REF → referring dept_id of Dept)
mgr, Salary (>5000),
comm (if not given default 100)
email
phone unique.

Page No.

Date: / /

```
CREATE TABLE EMPLOYEE(
emp_id NUMBER PRIMARY KEY,
emp_name varchar(30) NOT NULL
dept_id NUMBER REFRENCES Dept(dept_id),
job varchar2(30), mgr varchar(30)
salary NUMBER CHECK (salary >= 5000),
comm NUMBER DEFAULT 100,
email varchar(30),
phone varchar(12),
UNIQUE (email, phone));
```

## DB_INSERT.

Insert: It is used to insert data.

Syntax: In that order

```
INSERT INTO <table name>
values (attr. value1, attr value2. ...)
```

Eg:
```
INSERT INTO Dept
value (1, 'CSE', 'Hyderabad')
```

Syntax 2: In other order.

```
INSERT INTO <table name> (attr1, attr2, attr3 ...)
values (attr_value1, attr_value2 ...)
```

Page No.

Date __/__/____

# DB - delete and Update

Delete : Delete the data from exsisting table.

Syntax:

```
DELETE FROM <table name>
WHERE <condition>;
```

Delete * from Emp;
Emp.
(table exsists)

DROP TABLE Emp;
(table & data deleted)

Dept

| dept id (PK) | dept name | loc^n |
|---|---|---|
| 2 | ece | kgp |
| 3 | ecc | Hyd. |

Emp.

| Emp id | dept id (FK) | ename |
|---|---|---|
| 2 | 2 | def |
| 3 | 1 | ghi |

Delete from Dept
where dept id = 1; → Refrential integrity gets voilated

Drop; deletes both the data & table
Syntax : DROP TABLE <table name>

We can't use insert after drop.

Page No.

Date __/__/____

Update : Used to update the data.

Syntax:
```
UPDATE <table name>
SET <attr> = <value>
WHERE <condn> ;
```

Eg:
```
Update Emp.
SET dept_id = 1
WHERE emp_id = 2;
```

## DB - Referential triggered actions

Referential integrity gets hindered when you try to update or delete the data. So to maintain the referential integrity

| dept_id | dept name. |
|---|---|
| 1 | CSE |
| 2 | ECE |
| 3 | EEE |
| 4 | IT |
| 5 | CE |
| 6. | ME |

| emp_id | dep_id |
|---|---|
| 1 | 3 |
| 2 | 2 |
| 3 | 4 |
| 4 | 6 |
| 5 | 5 |
| 6 | 1 |

On delete

On update

1. Set Null
2. Set default
3. set Cascade

Page No. ____

Date __/__/____

It allows us to further describe the relationship between the ref. column and the object it references by attaching a 'referential triggered action' to the foreign key.

3 actions are
1) SET Null
2) SET Default
3) SET cascade

SET NULL

CREATE TABLE Emp (
Emp_id NUMBER,
Dept_id NUMBER REFRENCES Dept(dept_id)
ON DELETE SET NULL ON UPDATE SET NULL

Ex: DELETE FROM Dept
WHERE dept_id = 1;
UPDATE Dept
SET dept_id = 7
WHERE dept_id = 6;

SET Default (default value is never be deleted to maintain the integrity constant)

Ex: dept_id NUMBER DEFAULT 3 REFRENCES Dept(dept_id)
ON DELETE SET DEFAULT ON UPDATE SET DEFAULT

DELETE FROM Dept where dept_id = 2;
UPDATE Dept SET dept_id = 8 WHERE dept_id

Date __/__/____

SET CASCADE

dept_id NUMBER REFRENCES Dept(dept_id)
ON DELETE SET CASCADE ON UPDATE SET CASCADE

DELETE FROM Dept WHERE dept_id = 4;
UPDATE Dept SET dept_id = 9 WHERE dept_id = 5;

DB Alter

Alter: It is used to add, delete or modify columns in an existing table.

Also used to add and drop various constraints on an existing table.

Create table Emp( Emp_id NUMBER, emp_name Char(10)
UNIQUE, Salary NUMBER
check(salary >= 5000),
Phone varchar2(13),
pincode varchar(6));

ALTER TABLE Emp
ADD PRIMARY KEY(emp_id);

ALTER TABLE Emp
DROP UNIQUE (emp_name);

ALTER TABLE Emp
MODIFY CHECK (salary >= 1000);

Page No.

Date __/__/____

for attributes

ALTER table Employee
① ADD Add varchar(30);
① MODIFY phone varchar(20);
① DROP COLUMN pincode;

## DB_Select

Used to retrieve data from database.

Syntax: SELECT <attr_list> π
FROM <table_list> ×
WHERE <conditions>; σ

attr_list : list of attributes whose values are to be retrieved by the query.

table_list : list of tables from which we retrieve the data.

Condition: Conditional [Boolean] expressions that identifies the rows retrieved by query.

Dept

| dept_id | dep_name |
|---|---|
| 1 | CSE |
| 2 | IT |
| 3 | ECE |
| 4 | EEE |
| 5. | ME |

Emp

| emp_id | dept | emp_name | Job |
|---|---|---|---|
| 1 | 5 | Sindhu. | a |
| 2 | 2 | Srinivas. | b |
| 3 | 1 | Santosh | a |
| 4 | 2 | Saumya | a |
| 5 | 4 | Sindhu | b |
| 6 | 3 | Saumya | c |

Page No.

Q. Retrieve the names of all employee who work for dept 1.

```
Select emp_name
from Emp
where dept = 1;
```

Q. Retrieve the names, empid of the employees who work for IT dept.

```
Select emp_name, empid
from Emp, Dept
where dept = dept.id
AND dept_name = 'IT';
```

Q. Retrieve the jobs of all employee.

Select Job from Emp; {a, b, a, a, b, c}

Select distinct Job from Emp; {a, b, c}.

DB_view.

| dept_id | dept_name |
|---|---|
| 1 | CSE |
| 2 | ECE |
| 3 | ME |
| 4 | EEE |
| 5 | CE |

| Stud_id | stu_name | dept |
|---|---|---|
| 1 | Sindhu | 2 |
| 2 | Soumya | 1 |
| 3 | Srinivas | 3 |
| 4 | Santosh | 2 |
| 5 | Subbu | 4 |
| 6 | Sirarani | 2 |
| 7 | Satish | 2 |

Date __/__/__ → view

Create table student_ece as
(Select stu_name, stu_id
from student, Dept
where dept = dept_id AND
dept_name = 'ece');

Same data at multiple places → coherences

Dynamic table - view.

View : It is a virtual table based on the result set of a SQL statement.

Syntax: CREATE VIEW <view name> AS
(Select <cd name> FROM <table name>
where <condition>

## DB_Aliasing

SQL aliases are used to rename a table or a column in a table.

Q. For each employee retrieve employee's id, name, salary & the name of his/her immediate supervisor.

| Dept id | Dept_name |
|---|---|
| 1 | Accounting |
| 2 | Sales |
| 3 | Research |
| 4 | Finance |

Date __/__/____

Emp

| emp_id | dept_id | sup_id | Sal | emp_name |
|---|---|---|---|---|
| 1 | 1 | 7 | 11000 | Sindhu |
| 2 | 3 | 4 | 2200 | Soumya |
| 3 | 1 | 7 | 2100 | Santosh |
| 4 | 4 | 7 | 3000 | Satish |
| 5 | 2 | 4 | 2000 | Sanchana |
| 6 | 1 | 7 | 5000 | Murali |
| 7 | 1 | – | 1500 | Ravi |

```
Select E.emp_id AS "Employee id",
E.Sal AS "Salary",
E.emp_name AS "employee_name"
S.emp_name AS "Supervisor_name"
FROM EMP AS E, EMP AS S
WHERE E.sup_id = S.Emp id ;
```

O/P

| Employee id | Salary | employee_name |
|---|---|---|
| | | |
| | | |

Retrieve the id of employee and their corresponding name of dep.

```
Select E.emp_id AS "Employee Id",
D.dept_name AS "Department_name"
FROM Emp E , Dept D
WHERE E.dept id = D.dept id ;
```

Page No.

Date / /

# DB_Pattern Matching

LIKE : It is used to search for a specific pattern in a column.
We used 2 wild cards
% → represents any sequence of 0 or more characters
_ → used to replace a single character.

a % → first character in the name is a
_a% → second character in the name is a
%a → ending with a.
%a_ → last one char is a.

Q. Retrieve all the students whose name starts with R.

```
Select * from student
WHERE name like 'R%';
```

Q. Display the names of students who secured 4 digit rank.

```
Select name from student
WHERE rank like '____';
```

% as the second symbol use escape sequence to search for data field as a%.

```
Select name from student where marks like '90\%'
escape '\' and email like 'abc\_%@%.%'
escape '\';
```

NOT LIKE

where name NOTLIKE 'R%';

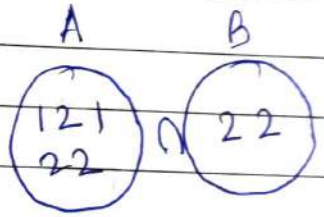
↓

name of students whose name doesn't start with R.

DB Set operations.

Union (∪)
Intersect (∩)
Except (−)
} duplicates are not allowed.

Unionall
Intersectall
Except all
} duplicates allowed.




Retrieve the list of students who got either grade 'A' or grade 'B'. (with duplicates)

```
Select stdId from Enroll
    where grade = 'A' ;
    Union All
Select stdId from Enroll.
    Where grade = 'B' ;
```

Union = (1, 2)
Union all = (1, 2, 1, 2, 2)

Both grade 'A' and grade 'B'.

```
Select StudId from Enroll where grade = 'A';
    Intersect All.
Select StudId from Enroll where grade = 'B';
```

Intersect = (2)
IntersectAll = (2, 2)

Date ___ / ___ / ____

who got grade 'A' but not grade 'B'.

Select StudId from enroll where grade='A';
Except
Select Stud Id from enroll where grade = 'B';

Except = { 1 }.
Except All = { 1, 1, 2 }.

## DB Airthmetic Operators

SQL allows use of airthmetic operators in queries on numeric domain.

Q. Increase the salary of an employee by 10% and display the salary and employee name.

Select Emp name, salary * 1.1 from Employee;

Q. CONCATENATE - for string datatype the concatenate operator '||' can be used in a query to append 2 string values.

Select Fname || Lname AS "FULL Name"
from Employee;

BETWEEN - It is a comparision operator on numeric datatype

Q. Retrieve all the employees whose salary is between 30,000 and 50,000.

Date __/__/____

Select * from Employee
where Salary Between 30000 AND 50000;

NOT BETWEEN - vice versa on between

Between can be used on text & dates.

Select * from Employee where Emp_Name
Between 'A' and 'D';

O/P A, AA, B, Ba, D, Dax
↓
greater than D not displayed.

In case of strings
Between = [A, D] Inclusive
NOT Between = (A, D) Exclusive. (B, C, D)

Select * from Employee where
hire_date Between '1-Jan-2010' AND '1-JUL-2010';

## DB - orderby

Order by is used to order or sort sql query in ascending or descending order.

Syntax: Select <column_list> FROM
<table_list> ORDER By
<column_1> ASC/DESC, <column2> ASC/DESC;

Date __/__/__

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

Select A,B,C from R Order By A, B, C ;

In Asc order 1 2 1 ← O/P.
1 2 3.
2 1 1
2 1 3.
3 4 3.
3 5 4

Select A,B,C from R. ORDER BY A DESC, B, C DESC

O/P → 3 4 3.
3 5 4 .
2 1 3
2 1 1
1 2 3
1 2 1

Order by Example

Q Retrieve the list of employee names, their dept, & project names they are working on ordered by dept name & with in each dept. order

alphabetically by last name, fname.

```
Select d.Dname, e.Lname, e.fname, P.name from
EMPLOYEE e, works_on w, PROJECT P, Department d
where d.number = e.Dno AND
      e.SSN = w.eSSN AND.
      w.pno = p.Pnumber;
Order by d.dname, e.Lname, e.fname;
```

| | Dname | Lname | fname | Pname |
|---|---|---|---|---|
| 1 | a | a | b | $P_1$ |
| 3 | b | a | c | $P_2$ |
| 2 | a | b | c | $P_1$ |
| 5 | c | c | b | $P_3$ |
| 4 | b. | c. | b. | $P_2$ |

DB_null values.

Dealing with null values at diff^n. context.
1) Arithmetic expressions (+, −, *, /)
2) Logical expressions

1)

| A | A+1 |
|---|---|
| 5 | 6 |
| 4 | 5 |
| Null | NULL |
| 1 | 2 |

Null + arithmetic op = NULL

2) 1 < NULL — unknown

(1 < NULL) AND (1 < 2)
UN T

Date / /

| A | B |
|---|---|
| T | T |
| T | F |
| F | T |
| F | F |
| T | UK |
| F | UK |
| UK | UK |

for null values

| A AND B |
|---|
| T |
| F |
| F |
| F |
| UK |
| F* |
| UK |

| A OR B |
|---|
| T |
| T |
| T |
| F |
| T* |
| UK |
| UK |

NOT(T) = F
NOT(F) = T
NOT(UK) = UK

DB Null values in various contexts

R

| A | B |
|---|---|
| a | 1 |
| b | 2 |
| C | NULL |
| d | NULL |

Select A
from R
where B = NULL ✗ o/p nothing
where B IS NULL ✓
o/p c, d

Where B IS NOT NULL
o/p → a, b

e

| N |
|---|
| a |
| b |
| C |
| NULL |
| NULL |

d

| N |
|---|
| a |
| b |
| C |
| NULL |
| NULL |

we don't get NULL value in join selection

Page No.

Date __/__/____

Using DISTINCT we can consider every NULL as same

R

| A | B |
|---|---|
| a | 1 |
| b | 2 |
| c | NULL |
| d | NULL |

Select B from R
1, 2, N, N

Select DISTINCT B from R
1, 2, N.

(1, 2, N, N) Union (1, 3, N, N)

INTERSECT

(1, 2, 3, N) → (1, N)

## IN Operator

The IN operator allows you to specify multiple values in a where clause.

Syntax: Select column_name(s) FROM Table name
WHERE Column_name IN (val1, val2 ...);

Ex: Retrieve the fnames of employees who works for the dept. 2, 3, 4, 5.

```
Select fname
From Employees
Where dept IN (2, 3, 4 5);
```

Page No.

Date ___/___/___

Q. Find the FName, Address of employees who work for the department located in 'New York'.

```
Select FName, Adress
From Employees
WHERE DNo IN
(Select DNo from Dept_location
where location = 'NEWYORK');
```

Subquery → correlated / non-correlated

↓

Non correlated

Inner table could be executed first

Q. Retrieve the Fname, Lname of all managers.

```
Select FName, Lname
From Employee e
WHERE SSN IN
(Select MgrSSN from Department d);
```

Q. Retrieve the FName of each employee who has a dependent with the Fname & same Sex as the employee

Correlated

```
Select e.FName FROM EMPLOYEE e
WHERE e.SSN IN (Select essn from dependent d
WHERE e.Fname = d.dependent name AND e.sex = d.sex
```

Page No.

Date __/__/____

8. Find out all the employee id's who worked on same (or) any project on which employee 10 has worked for same number of hours.

```
Select distinct essn from WORKS_ON
WHERE (Pno, hours) IN
(Select pno, hours from WORKS_ON
 WHERE essn = 10);
```

| essn | Pno | Hours |
|---|---|---|
| → 10 | 20 | 100 |
| 20 | 20 | 100 |
| 30 | 10 | 10 |
| 40 | 20 | 100 |
| → 10 | 30 | 20 |

| | |
|---|---|
| 20 | 100 |
| 30 | 20 |

10, 20; 40, 10 ⇒ 10, 20, 40

DB- Any/Some, All (>, <, >=, <=, <> =)

9. Find out fnames of all the employees whose salary is greater than all employees in Dno = 5.

```
Select Fname from Employee
WHERE Salary > ALL( Select Salary from
  employee where Dno = 5);
```

ALL a, 11K (10K, 20K, 30K) ✗
SOME/ANY a, 11K (10K, 20K, 30K) ✓

= any / = some ⇒ in

Page No. ____

Date __/__/____

= any b, 10K (10K, 20K, 30K)

• in IN we can compare more than 2 values

→ only one value can be compared

<> (NOT Equal to)

<> some ≅ <> Any → not equal to any

≅ NOT IN

## DB – Exists / NOT Exists

The SQL exists condition is used in comb$^{n}$ with a subquery and is considered to be met if the subquery returns atleast 1 row.

Q. Retrieve the fnames of employees who have dependents with same fname, sex as that of the employee.

e

| SSN | FName | Sex |
|---|---|---|
| 10 | a | M |
| 30 | b | F |

d

| essn | Sex | dept |
|---|---|---|
| 10 | M | a |
| 30 | F | b |

```
Select fnames from Employee e
WHERE EXISTS (select * from
Dependent d WHERE e.ssn=d.essn
AND e.sex=d.sex AND e.fname=d.dept
```

Q. Retrieve the fnames of the employee who have no dependent.

```
Select fname from Employee e
WHERE NOT EXISTS ( select * from Dependent
where SSN= e.ssn)
```

# DB - Examples on Exists & Not exists

List the Fname of managers who have atleast one dependent.

```
Select Fname from Employee e
WHERE EXISTS (Select * from
dependent d where e.SS = d.essn
    AND EXISTS (Select * from Department
    Dept where e.SSn = Dept.Mgrssn);
```

Employee

| Fname | SSN |
|---|---|
| a | 10 |
| b | 20 |

Department

| Deptno | Mgrssn |
|---|---|
| | 10 |

Dependent

| essn |
|---|
| 10 |
| 20 |

(1) He is manager or not
(2) He has dependent or not

Q. Retrieve the fname of each employee who works on all the projects controlled by dept number 5.

```
Select Fname from Employee e
WHERE exists (Select Pno from WORKS_ON
    (Select Pnumber from Project
    WHERE Dnum = 5)
    EXCEPT
    (Select Pno from WORKS_ON w Where
    e.SSn = w.essn);
```

A - B

Date __/__/____

## Non correlated query

Employee

| Fname | SSN |
|---|---|
| a | 1 |
| b | 2 |

Project

| PNO | dNO |
|---|---|
| 10 | 5 |
| 20 | 1 |
| 30 | 5 |
| 25 | 5 |

Works on

| Pno | ESSN |
|---|---|
| 10 | 1 |
| 15 | 1 |
| 25 | 1 |
| 30 | 1 |
| 40 | 2 |

Select Fname from Employee e where NOT EXISTS

(10, 15, 25) − (10, 15, 25) returns True

(10, 15, 25, 30) − (40)

returns false.

O/p → a.

## DB Aggregate function

Set → no duplicates

multiset → duplicates

Aggregate functions are functions that take a collection (a set or multiset) of values as i/p & return a single value.

e) AVG : Average
MAX : Maximum
MIN : Minimum
SUM : Total
COUNT : Count

Date / /

Select AVG (salary)
from emp;

Select COUNT (salary) → how many salary are there.
from emp;

AVG (MARKS_A + MARKS_B) ✓

WHERE SUM X not allowed where with aggregate func^n's

Dealing with NULL values in Aggregate func^n.

All aggregate func^n. except count(*) ignore NULL values in their input collection.

* The count of an empty collection is defined to be zero(0) & all other aggregate oper^n's return a value of NULL when applied on an empty set.

| A | B |
|---|---|
| 1 | N |
| 2 | N |
| N | N |
| 3 | N |

null ignored

count(*) = 4
count(A) = 3
count(B) = 0

Sum(A) = 6
Avg(A) = 2
MAX(A) = 3
MIN(A) = 1

SUM(B) = N
AVG(B) = N
MAX(B) = N
MIN(B) = N

N → null.

Date __/__/____

# DB_ Group By clause.

Q. For each department that has more than 5 employees retrieve the department name & the number of its employees who are making more than 10,000.

```
Select Dname, count(*)
FROM Department, Employee WHERE
Dnumber=Dno and Salary > 1000 AND
Dno IN ( Select Dno from Employee
         GROUP BY Dno Having count(*)>5)
         GROUP BY Dname;
```

Q. Find the avg salary in each dept.

```
Select Dno, AVG(salary)
FROM Employee
GROUP BY DNo;
```

Employee

| SSN | Dno | Sal |
|---|---|---|
| 10 | 1 | 100 |
| 20 | 2 | 200 |
| 30 | 3 | 300 |
| 40 | 1 | 400 |
| 50 | 2 | 500 |
| 60 | 3 | 600 |

| | |
|---|---|
| 1 | 100 |
| 1 | 400 |
| 2 | 200 |
| 2 | 500 |
| 3 | 300 |
| 3 | 600 |

| | |
|---|---|
| 1 | 250 |
| 2 | 350 |
| 3 | 450 |

Date ___/___/___

All attributes used in GROUP BY clause need to appear in the Select clause. Any attribute that is not present in GROUP BY clause must appear only inside the aggregate in the select clause.

HAVING clause → for filtering group

SF W (G) (H)

Only Group by is there, then there is Having.

Ex: List the dpno with AVG salary > 20,000.

```
Select Dno, Avg(salary)
FROM employee
GROUP BY DNO.
HAVING AVG(salary) > 20,000;
```

DB- Gate 2012.

Consider the following rel^n. 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 | Age |
|---|---|---|
| 10 | 2200 | 02 |
| 99 | 2100 | 01 |

How many tuples does the result of the following SQL query contain?

```
Select A.Id from A WHERE A.Age >
A.A (select B.Age FROM B WHERE
        B.NAME = 'Arun');
```

Page No. ____

Date ___/___/_____

B

| Age |
|---|

B

ALL (Empty set)

| A |
|---|
| 12 |
| 15 |
| 99 |

Ans: 3

DB_With Clause

The SQL "with" clause allows you to give a sub-query block a name, a process also called sub query factoring which can be refrenced in several places within the main SQL query.

The name assigned to sub-query is treated as though it was a view or table. It is basically a drop in replacement to the normal sub query.

Ex: Select emp_id from Employee;

Using with: WITH Emp_tab AS( Select empid from Employee)

Select * from Emp_tab;

Multiple WITH clauses:

WITH Dept_temp AS( Select dept id from Departments WHERE d name='CSE'),

Emp_name AS( Select emp name from employee E, Dept_temp D WHERE E.deptid = D.dept_id

Page No.

8. For each employee, display the number of employees working in their respective department.

```
Select e.ename AS emp_name, DC.dept_count AS
emp_dept_count FROM emp e,
(select dept no, count(*) AS dept_count
FROM emp GROUP BY dept no) DC
WHERE e.dept no = DC.dept no;
```

WITH:

```
WITH dept_count AS (Select deptno, count(*) AS dept_count
                    from emp GROUP BY dept no)

Select e.ename AS emp_name, DC.dept_count AS
emp_dept_count
FROM emp e, dept_count DC
WHERE e.dept no = DC.dept no;
```

Joins

1) Normal Join

```
Select * from (R JOIN S ON A=C)
```

R

| A | B |
|---|---|
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |

S

| C | D |
|---|---|
| | |

Date ___/___/_____

Normal Join takes place when we provide any condition.

R

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

S

| C | D |
|---|---|
| 1 | g |
| 2 | h |
| 3 | i |
| 7 | j |
| 8 | k |
| 9 | l. |

Select * from (R JOIN S ON A=C)

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

2) Natural Join : Natural join doesnt have condition. So natural join only takes place when there is an matched attribute

Rename C as A

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

Page No.

Date ___/___/___

3. Left outer Join: Join 2 tables add whatever in left table.

| A | B | C | D |
|---|---|---|---|
| 1 | a | 1 | g |
| 2 | b | 2 | h |
| 3 | c | 3 | p |
| 4 | d | N | N |
| 5 | e | N | N |
| 6 | f | N | N |

4. Right Outer Join: Join 2 tables add whatever is right table.

| A | B | C | D |
|---|---|---|---|
| 1 | a | 1 | g |
| 2 | b | 2 | h |
| 3 | c | 3 | i |
| N | N | 7 | j |
| N | N | 8 | k |
| N | N | 9 | l |

5 Full outer Join: Add whatever present on left as well as right side.

# Difference between Alter & Update

| Alter | Update |
|---|---|
| 1) It is a DDL command | 1) It is DML command |
| 2) It works on table structure | 2) It works on table data |

```
update Emp
Set salary = salary * 2 ;
Where name = "A";
```

Emp

| id | name | Salary | Email |
|---|---|---|---|
| 1 | A | 10000 | |
| 2 | B | 20000 | |
| 3 | C | 30000 | |

20000
40000
60000

# Difference between delete drop, truncate

(Logs are there)

Rollback ✓ | Rollback ✗ | (no logs are there)

| Delete | Drop | Truncate |
|---|---|---|
| DML | DDL | DDL |
| Delete from Student; (Every row is deleted) Delete from Student Where id="1"; | Drop table Student (whole structure/ schema is deleted) | Truncate Student; (It deletes every row) (At once in go delete every row) |

Student

| id | name | |
|---|---|---|
| 1 | A | x |
| 2 | B | x |
| 3 | C | x |

Rollback Student ✗

Delete has an option to apply condition so its slow but truncate is fast

Constraints in SQL
(Condition or restrictions on DB)

Conditions are put upon columns.

(1) Unique (no duplicate values)

2) NOT NULL ⇒ User can't leave the column empty

3) (Unique + Not Null) Primary key - The column should be unique as well as not null

4) Check - Before entering the value it checks the value.

5) Foreign key - Used to maintain refrential integrity.

6) Default → In case no data, there is any value.

| 4 | Nitin | | |
|---|---|---|---|
| 5 | Varun | IT | 50 000 |

SQL queries & Sub queries

Q1. Write a SQL Query to display max^m. salary from Emp.

Q2. Write a SQL Query to display Employee name who is taking maximum salary.

1) Select MAX(salary) from Employee;

2) Select Ename from Employee
Where salary = (Select MAX(Salary)
from Emp);

Q3. Write a SQL to display the second highest salary from the table.

Select Max(Salary) from Employee
Where salary <> (Select Max(Salary)
from Employee);

Q4) Write a SQL query to display Employee name who is taking 2nd highest salary.

Select Ename from Employee
Where Salary = (Select Max(salary) from
Employee where salary <> (Select Max(Salary)
from employee));

= → Only 1 value  2=2  T

IN → for multiple values  2 IN{1, 2, 3, 4, 5}  T

## Group By (Solving by groups)

Q. Write a query to display all dept names along with no of emp working in that?

o/p →

| | |
|---|---|
| HR | 2 |
| MRKT | 2 |
| IT | 1 |

group by → aggregate func^n.

```
Select dept, count(*) from Emp
Group by dept;
```

## Having (It works as where in group by since where works on whole table)

Q. Write a query to display all the dept names where no. of employees are less than 2.

```
Select dept from Emp
group by dept having count(*) < 2;
```

o/p → IT

Q. Write a query to display highest salary department wise name of emp who is taking that salary..

Select E_name from Emp
where Salary IN
(Select max(Salary) from Emp
group by department);

## IN & NOT IN

Q

Emp

| Eid | Ename | Address |
|---|---|---|
| 1 | Ravi | Chd |
| 2 | Varun | Delhi |
| 3 | Nitin | Pune |
| 4 | Robin | Banglore |
| 5 | Ammy | Chd |

Project

| Eid | Pid | Pname | location |
|---|---|---|---|
| 1 | $P_1$ | IOT | Banglore |
| 5 | $P_2$ | BIG Data | Delhi |
| 3 | $P_3$ | Retail | Mumbai |
| 4 | $P_4$ | Android | Hyderabad |

find the detail of Emp whose address is either Delhi or chd or Pune.

Select * from Emp
WHERE Address IN ('Delhi', 'Chd', 'Pune');

Not from Delhi & Pune

```
Select * from Emp
where Address NOT IN ('DELHI','PUNE');
```

Q. find the name of Emp who are working on a project.

```
Select ename from Emp
WHERE Eid IN

(Select distinct (eid) from project)
```

O/P
Ravi
Nitin
Robin
Ammy

EXIST AND NOT EXSIST

~~Nested~~ Correlated Querys

Nested Query — Outer — IN, NOT IN, ~~ANY ALL~~ — Inner

Correlated Query — outer — Exists, Not exists — Inner

Exists

Find the detail of Emp who is working on atleast one project.

```
Select * from Emp
where Eid exists ( Select Eid from project
  where Emp.eid = Project.Eid)
```

Top down approach

1 row from outer → In inner one

Inner query repeats every time till outer query.

## SQL Aggregate functions
(SUM, AVG(n), COUNT, MIN, MAX)

Emp

| Eid | Ename | Dept | Salary |
|---|---|---|---|
| 1 | Ram | HR | 10 000 |
| 2 | Amrit | MRKT | 20 000 |
| 3 | Ravi | HR | 30000 |
| 4 | Nitin | MRKT | 30000 |
| 5 | Varun | IT | 50 000 |
| 6 | Sandy | Testing | Null |

(1) Select MAX(salary) from Emp; o/p 50000

(2) Select MIN(salary) from Emp; o/p 10000

(3) Select count(*) from Emp;

o/p: total no of tuples o/p → 6

(4) Select count(salary) from Emp;

o/p → 5 (doesn't counts nulls)

(5) Select distinct (count(salary)) from Emp;

o/p → 4

(6) Select SUM(salary) from Emp;

o/p → 140000

(7) Select Distinct SUM(salary) from Emp;

o/p → 100000

(8) Select Avg(salary) from Emp;

o/p = 140000/5

Distinct Avg = $\frac{\text{Distinct SUM}}{\text{Distinct count}}$

Q. Find the details of all employee who works in department.

takes less time — takes more time

| Joins | Nested Subquery | Correlated Subquery |
|---|---|---|
| Cross product + condition | Bottom up | Top down approach |
| Select * from Emp, dept where emp.eid = dept.eid | Select * from Emp WHERE eid IN (Select eid from dept) | Select * from emp. where EXISTS (Select eid from dept WHERE emp.eid = dept.eid); |

Emp

| Eid | name |
|---|---|
| 1 | A |
| 2 | B |
| 3 | C |
| 4 | D |
| 5 | E |

Dept

| Dept no | Name | Eid |
|---|---|---|
| $D_1$ | IT | 1 |
| $D_2$ | HR | 2 |
| $D_3$ | MRKT | 3 |

Find the Nth highest salary using SQL.

```
Select id, salary from Emp e1
    where N-1 = (Select Count (distinct salary)
                 from Emp e2
                 where e2.salary > e1.salary);
```

# Basic PL SQL Execution Part 1

Program 1: Find the sum of 2 numbers

```
declare
 a int;
 b int ;
 c int ;
 begin
 a:= &a;  // (live server doesn't work but on system server does)
 b:= &b;
 c:= a+b;
 dbms_output.put_line ('sum of a and b=' || c);
 end;
```

Program 2: Greeatest of 2 numbers

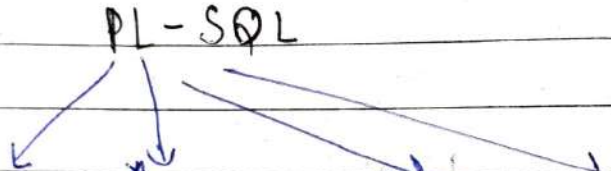
```
declare
 a int;
 b int;
 begin
 a:= &a;
 b:= &b;
 if (a>b)
 then
 dbms_output.put_line ('a is greater' || a);
 else
 dbms_output.put_line ('b is greater' || b);
 end if;
 end;
```

Emp e1

| Id | Salary |
|---|---|
| 1 | 10 |
| 2 | 20 |
| 3 | 20 |
| 4 | 30 |
| 5 | 40 |
| 6 | 50 |

Emp e2

| Id | Salary |
|---|---|
| 1 | 10 |
| 2 | 20 |
| 3 | 20 |
| 4 | 30 |
| 5 | 40 |
| 6 | 50 |




SQL → declarative (what to do)

PL SQL → procedural (what to do + How to do)

| Block | Declaration a int |
|---|---|
| | Executable code: begin a := 10; end; |
| | Exception handling (error) |

# PL-SQL (while, for loop)

## For loop

```
Declare
 a number(2)
 begin
for a in 0..10
 loop
 dbms_output.put_line (a);
 end loop;
 end;
```

## while loop

```
declare
 a int;
 b int;
 begin
 a:= 0;
 b:= &b;
 while (a < b)
  loop
   a:= a+1;
 dbms_output.put_line(a)
 end loop;
  end;
```

## Single row & Multi row functions in SQL

- Single row (Single row Single O/p)
- Multi row (Multiple row Single O/p)

# Single Row function.

(A) Number func$^{n}$.
1) Round
2) Truncate
3) Mode.

(B) Character functions.

Case manipulation
1) Lower
2) Upper.
3) init cap.

Character manipulation
1) Concat
2) Substr
3) length
4) Instr
5) LPAD/RPAD
6) TRIM
7) REPLACE

(C) Date function
1) Month between
2) Add months
3) NEXT_DAY
4) LAST_DAY.
5) ROUND
6) TRUNC.

D) General functions.
i) NVL
ii) NVL 2
3) NULL IF
4) COALESCE

5) CASE
7) DECODE

5) Data type conversion
1) TO_CHAR
2) TO_NUMBER
3) TO_DATE

Multirow function
A) Aggregate

1) SUM
2) MAX
3) MIN
4) COUNT
5) AVG

Various char functions

Case function manipulation.

1) lower
Select lower(name) from emp;

2) Upper
Select upper(name) from emp;

3) Init cap.
Select initcap(name) from emp;

(B) Character manipulation.

1) Concat ('Varun' 'Singla') Varun Singla.

Select concat (id, name) from emp;

2) Substr ('Varun', 2, 4) aru.

Select substr (name, 2, 5) from emp;

3) Instr ('Varun', 'u') 4th.

Select Instr (name, 'T') from emp;

4) Length ('Varun') 5.

Select length (name) from emp;

5) Lpad ('Varun', 10, '*') *****Varun

Select lpad (name, 15, '*') from emp;

6) Rpad ('Varun', 10, '*') Varun*****

Select rpad (name, 15, '*') from emp;

7) Trim ('V' from 'Varun')

Select trim ('v' from name) from emp;

8) Replace ('Varun', 'V', 'T')

# Transaction And Concurrency control:

Transaction: Transaction is a collection of related Read & write operation used to perform some related task.

```
R(A)
A = A+500
W(A)
R(B)
B = B-500
W(B)
Commit
```

R(A) - Read operation
W(A) - write operation.

## ACID

Atomicity - (All or nothing) → Either execute all the operations of the transaction or none of them

Component of Database Responsible for it:

1) Recovery management Component → It takes care that either the atomicity is applied else Rollback is done in case of error arises before commit.

How?

1) It maintains a log file till commit.

2) Consistency → (No violation of integrity constants)

Rule → "If the Database was consistent before a particular transaction, then it should also be consistent after that execution of the transaction.

Who is responsible?
→ It is users / programmers responsibility

How does it do?
By setting various integrity.

3) Isolation
Concurrent changes are invisible
Rule → "Concurrent Execution of 2 or more transactions should not cause any inconsistency. It should be as if the transaction executed independent of other.

| $T_1$ | $T_2$ |
|---|---|
| R(A) → | context switch. ($\because T_1 \to T_2$) |
| W(A) | R(A) ($T_2 \to T_1$) |
| | W(B) |
| R(B) | |
| W(B) | |

4) Durability - (The commited Update Persists)
The effect of a completed or committed transaction should be persist even after a crash.

It means once a transaction commits, then system must guarantee that the result will never be lost.

How?
1) By ensuring Recoverability
2) By using RAID → It keeps multiple copies of information in various databases.

R → Redudant
A → Array of
I → Independent
D → Disks.

## Serializability : Basics & Classification

A Time order sequence of 2 or more transactions

Ex 1: S1

| $T_1$ | $T_2$ | $T_3$ |
|---|---|---|
| R(A) | | |
| | R(A) | |
| | | R(A) |
| W(B) | | |

S2

| $T_1$ | $T_2$ | $T_3$ |
|---|---|---|
| R(A) | | |
| | R(A) | |
| | | R(A) |
| W(B) | | |
| | W(B) | W(B) |
| commit | commit | commit |

S2 → complete schedule (commit operⁿ.)

Schedule

Serial

If only after the commit of one transaction, the other transaction begins then the schedule is said to be serial.

It satisfies all ACID prop.

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

Advantages
→ No inconsistency
Disadvantages
→ Poor throughput
→ Poor resource utilization

Concurrent

A schedule consisting of interleaved execution of 2 or more transaction simultaneously is called concurrent schedule.

| $T_1$ | $T_2$ | $T_3$ |
|---|---|---|
| R(A) | | |
| | R(A) | R(A) |
| W(B) | | |
| | W(B) | |
| | | W(B) |
| | | Commit |
| | Commit | |
| Commit | | |

Advantages
i) Better throughput
ii) Better source utilizat

Disadvantages
i) Inconsistency

Soln to the serializabilization?
→ Serializibity.

Serializability: Procedure & Conflict serializability

The problem of Inconsistency may arise in case of concurrent schedule.
Soln → serializability.

Convert the concurrent schedule into an equivalent schedule which has a "serial order execution of its constituent transactions"

Concurrent schedule → equivalent serializable schedule
↓
is consistent

How to convert concurrent schedule to consistent concurrent schedule

1) Conflict serializable
2) View serializable.

Conflict serializable.
A schedule is said to be "conflict serializable. if one of its conflict equivalent schedule is serializable.

Conflict Equivalent schedule ⇒ If S' is a schedule formed after swaping the non conflict pairs in a given schedule S, then S & S' are called equivalent schedule.

Conflict pairs

| | | | | | |
|---|---|---|---|---|---|
| R(A) | W(A) | W(A) | R(A) | W(A) | W(A) |

O/P

A pair which change if the order of the execution of these conflict pair is inverted.

Non conflict pairs

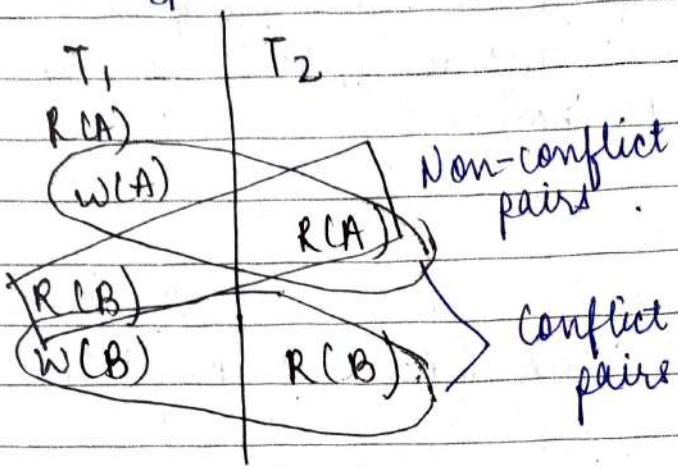
| | | | | | |
|---|---|---|---|---|---|
| R(A) | R(B) | R(A) | R(A) | W(A) | W(B) |

| | |
|---|---|
| R(A) | W(B) |

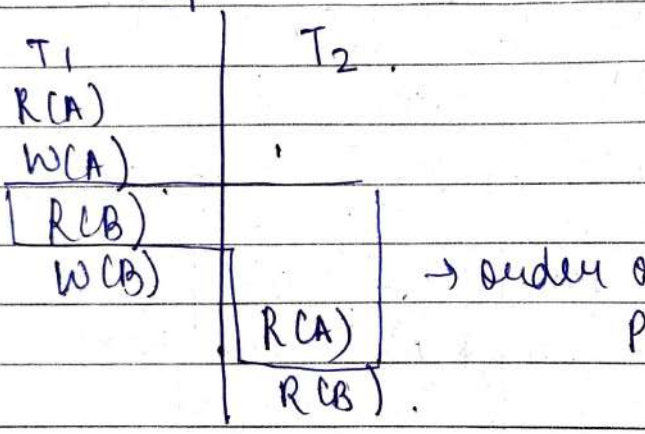
We can never swap the order of conflict pairs. As by doing so, the final result changes & the schedules no more remain equivalent

Finding a conflict Equivalent schedule

$S_1$




$S_1'$



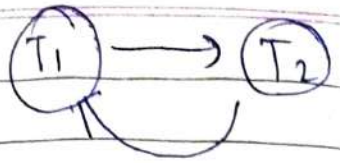

↙ Order: $T_1 \rightarrow T_2$.

Conflict Equivalent Schedule.

Precedence Graph.

Example 1: $R_1(x)$ $R_1(y)$ $R_2(x)$ $R_2(y)$ $W_2(y)$ $W(x)$ –

$R_1(y) \rightarrow W_2(y)$




$R_2(x) \rightarrow W_1(x)$

Cyclic
Hence
non-conflict
serializable.

Ex-2 $R_1(x)\ R_2(x)\ W_2(y)\ R_1(y)\ W_1(x)$

$R_2(x) \rightarrow W_1(x)$

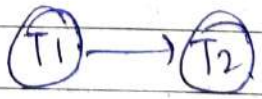

$W_2(y) \rightarrow R_1(y)$

Acyclic : conflict serializable

Order: $T_2 : T_1$

Precedence graph method.
node → transactions
edges → various conflicts

| $T_1$ | $T_2$ |
|---|---|
| R(A) | |
| | W(A) |




Acyclic : conflict serializable
Cyclic : Non-conflict serizable

In case the graph is Acyclic

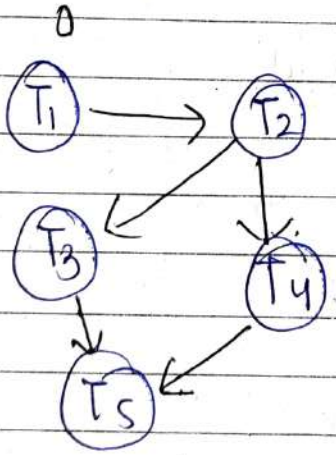
1) It is conflict serializable

We find the serial Execution Order as follows.

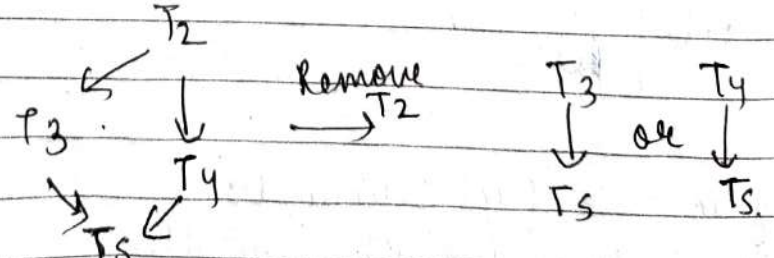
1) Remove the node with an Indegree 0 & also remove all the edges corresponding to it

2) Repeat Step 1 till all the Nodes are covered.

3) The final order of Execution of transactions is the order in which the nodes are removed from the graph.




↓ Remove $T_1$



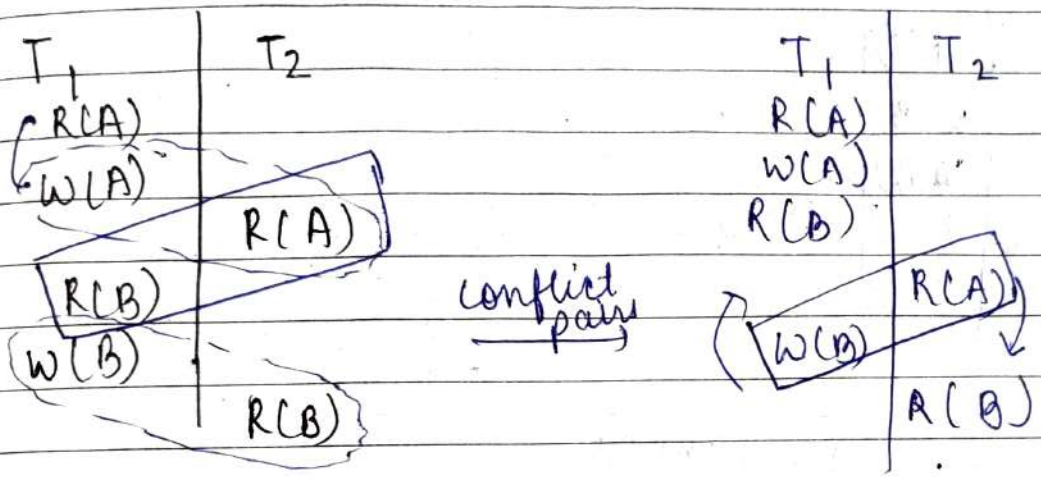

Serial order of schedule.

order 1 : $T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4 \rightarrow T_5$

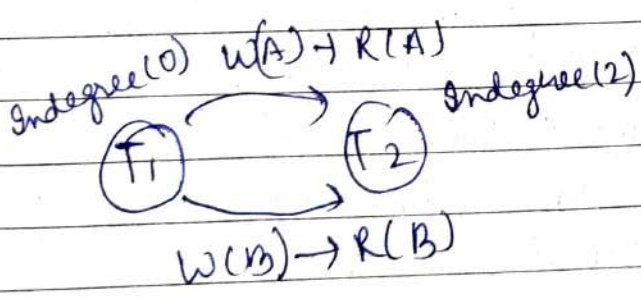
order 2 : $T_1 \rightarrow T_2 \rightarrow T_4 \rightarrow T_3 \rightarrow T_5$

# A Tabular form Example

Schedule
$S_1$



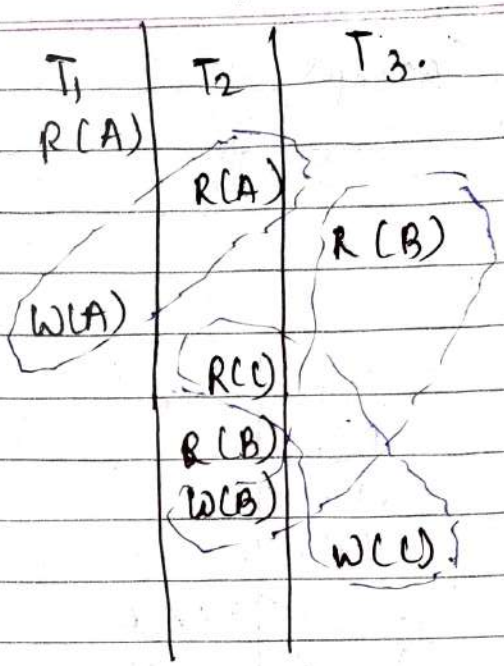

Order $T_1 \rightarrow T_2$

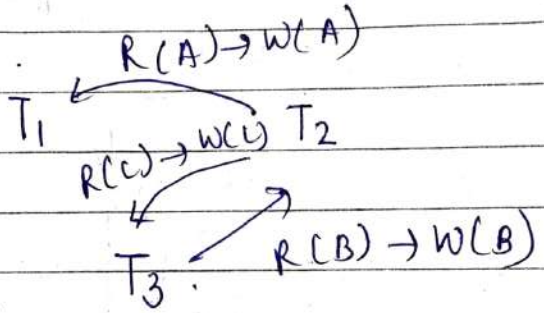



| $T_1$ | $T_2$ |
|---|---|
| R(A) | |
| W(A) | |
| R(B) | |
| W(B) | |
| | R(A) |
| | R(B) |

Acyclic ∴ Order $(T_1 \rightarrow T_2)$

$S_1$




Cycle → Non serializable.

Through conflict we only get to know it is non conflict serializable not non serializable.

View Serializability

Serializability

↓

Conflict serializable —Yes→ Serializable

↓ No

View serializable —Yes→ Serializable

↓ No

Non-serializable.

View serializable : A schedule is said to be view serializable, if it has an equivalent View equivalent schedule.

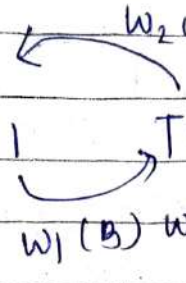
View equivalent schedule: A schedule S' is said to be view serializable to S if it satisfies the following 3 condn.

(1) Initial Read → If $T_1, T_2, T_3 \cdots T_j$ transaction perform initial read (i.e. Read the values of data items directly from the database before any modification (write operation) on the data items A, B, C ... J respectively in a schedule S then none of these Initial Reads should be violate in its view equivalent schedule S'.

(2) Read write sequence - The Read write conflict pairs should be in the same sequence in both the view equivalent schedules

(3) Final update → If Ti is a transaction that updates a data item (X) finally (at the end) in schedule S, then Ti should only update finally the data item X in the view equivalent schedule S'.

Eg: S: $R_2(B)\ W_2(A)\ R_1(A)\ R_3(A)\ W_1(B)\ W_2(B)\ W_3$




$T_3$

Non conflict serializable

there is a write first

| | A | B | |
|---|---|---|---|
| Initial Read | X | $T_2$ | $T_2 \rightarrow T_3$ |
| Final update | $T_2$ | $T_3$ | $T_2 \rightarrow T_1 \rightarrow T_3$ |
| Update | $T_2$ | $T_1, T_2, T_3$ | |

Order $T_2 \rightarrow T_1 \rightarrow T_3$

View Serializability: A tabular form

| $T_1$ | $T_2$ | $T_3$ |
|---|---|---|
| R(A) | | |
| | R(A) | |
| | | R(B) |
| W(A) | | |
| | R(C) | |
| | R(B) | |
| | W(B) | |
| | | W(C) |

$R_1(A)$ $R_2(A)$ $R_3(B)$ $W_1(A)$ $R_2(C)$ $R_2(B)$ $W_2(B)$ $W_3(C)$

| | A | B | C |
|---|---|---|---|
| Initial read | $T_1, T_2$ | $T_3, T_2$ | $T_2$ |
| update | $T_1$ | $T_2$ | $T_3$ |
| final update | $T_1$ | $T_2$ | $T_3$ |

A : $T_2 \rightarrow T_1$

B : $T_3 \rightarrow T_2$

C : $T_2 \rightarrow T_3$

Conflicting

So, no order can be there. So it is non-serilizable.

$R_1(A)$ $R_2(A)$ $R_3(B)$ $W_1(A)$ $R_2(C)$ $R_2(B)$ $W_2(B)$ $W_1(C)$

| $T_1$ | $T_2$ | $T_3$ |
|---|---|---|
| R(A) | | |
| | R(A) | |
| | | R(B) |
| W(A) | | |
| | R(C) | |
| | R(B) | |
| | W(B) | |
| W(C) | | |

| | A | B | C |
|---|---|---|---|
| Initial Read | | | |
| update | $T_1$ $T_2$ $T_1$ | $T_3$ $T_2$ $T_2$ | $T_2$ $T_1$ |
| final update | $T_1$ | $T_2$ | $T_1$ |

A : $T_1 \rightarrow T_2$

B : $T_3 \rightarrow T_2$

C : $T_2 \rightarrow T_3$

Page No.
Date

Combining all orders

$T_3 \rightarrow T_2 \rightarrow T_1$ $\quad T3 : T_2 : T_1$

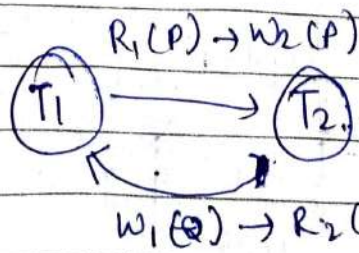
Gate 2012 (Serializability)

$T_1$ : Read (P)
Read (Q)
If P = 0 then Q := Q + 1;
write (Q)

$T_2$ : Read (Q)
Read (P)
If Q = 0 then P := P + 1;
write(P);

Any non-serial interleaving of $T_1$ & $T_2$?

| $T_1$ | $T_2$ |
|---|---|
| $R_1(P)$ | $R_2(Q)$ |
| $R_1(Q)$ | $R_2(P)$ |
| $W_1(Q)$ | $W_2(P)$. |




Loop is occurring
non conflict
serializable.

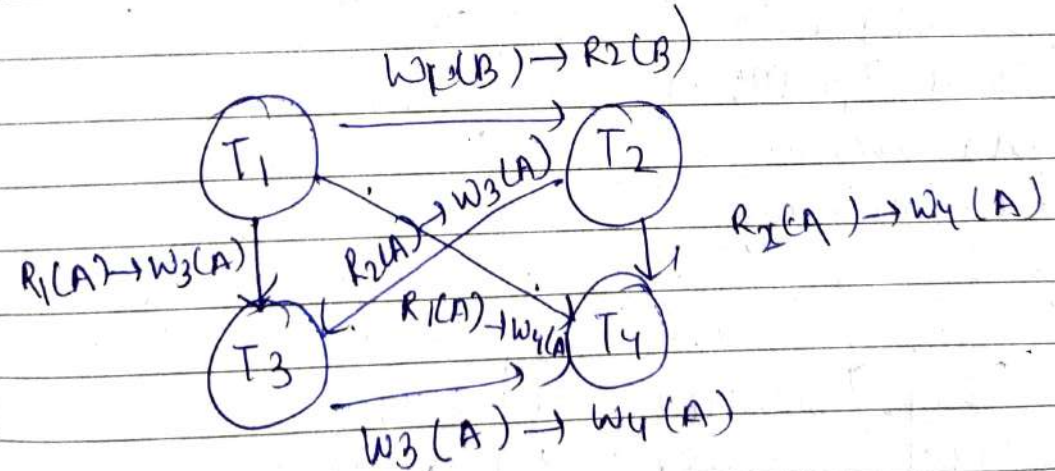
$R_1(P)\ R_4(Q)$

| | P | Q |
|---|---|---|
| Initial Read | $T_1, T_2$ | $T_1, T_2$ |
| Update | $T_2$ | $T_1$ |
| Final Update | $T_2$ | $T_1$ |

$T_1 \rightarrow T_2$ $\quad$ $T_2 \rightarrow T_1$.

Not view serializable.

Q: $R_2(A)$ $R_1(A)$ $W_1(C)$ $R_3(C)$ $W_1(B)$ $R_4(B)$ $W_3(A)$ $R_4(C)$ $W_2(D)$ $R_2(B)$ $W_4(A)$ $W_4(B)$.




Acyclic graph → serializable

order: $T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4$.

Check whether view serializable.

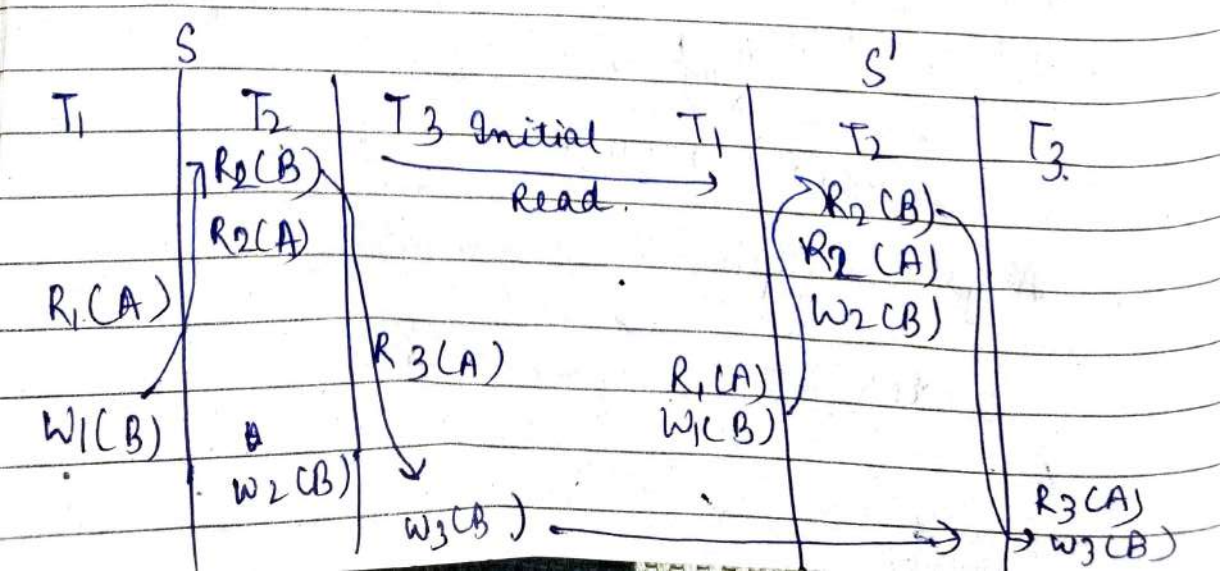
S: $R_2(B)$, $R_2(A)$ $R_1(A)$ $R_3(A)$ $W_1(B)$ $W_2(B)$ $W_3(B)$

| | A | B |
|---|---|---|
| Initial Read | $T_2, T_1, T_3$ | $T_2$ — (1) |
| update | X | $T_3, T_2, T_1$ — (2) |
| final update | X | $T_3$ → (3) |

Since no transaction is performing any write operation on Data item "A" So we can just ignore the operations on data item A for the time being

By (1) & (2) & (3)

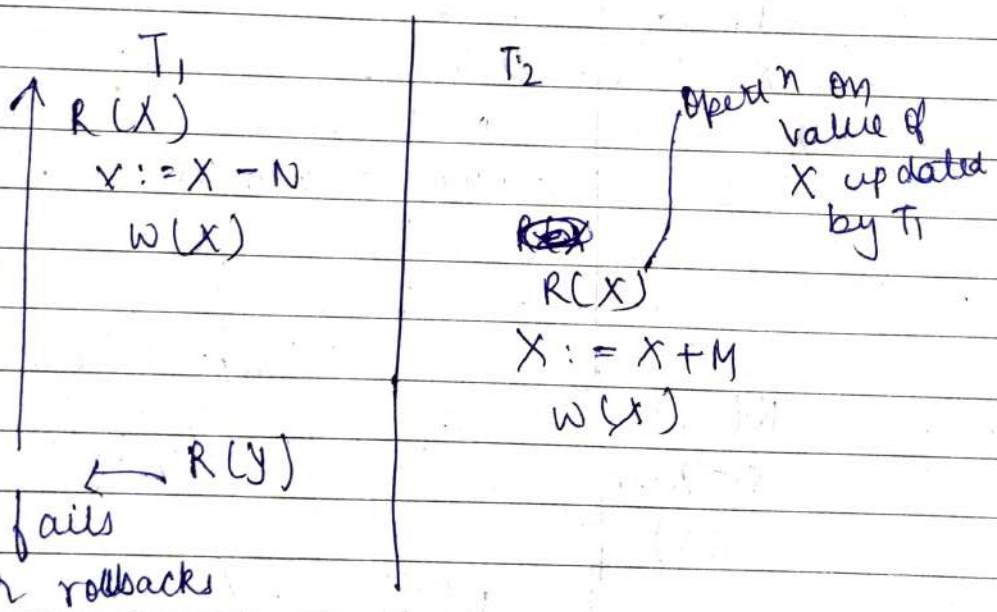
$T_2 \rightarrow T_1 \rightarrow T_3$



Q. ~~Check whether the schedule given is view serializable or not.~~

## Dirty Read Problem (W-R problem)

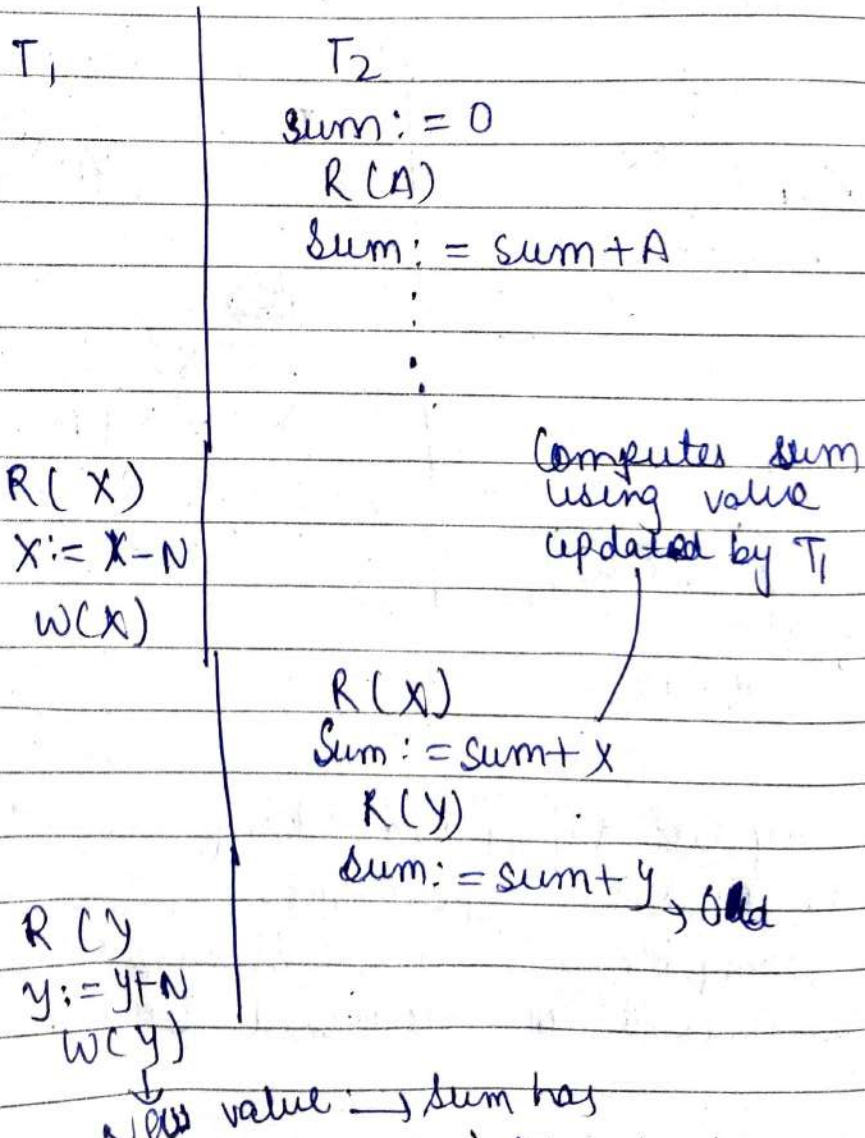
This problem occurs when one transaction updates a database item & then the transaction fails for some reason. Meanwhile the update item is accessed (read) by another transaction before it is changed back (or roll back) to its original value.




So, update by $T_1$ was temporary (W(X)) & the operations performed by $T_2$ on this temporary value should now be rollbacked or reversed back.

# The Incorrect Summary Problem.

If one transaction is calculating an aggregate summary function on a number of database items while other transactions are updating some of these items, the aggregate function may calculate some values before they updated & others after they are updated. So, the aggregate sum in such a case would contain a wrong value.



## The Unrepeatable Read Problem

Lets suppose a transaction T reads the same item twice & the item is changed by the another Transaction T' betⁿ. the two reads. Hence T receives diffⁿ. value for its 2 reads of the same item.

Example: During an airline reservation system, transaction a customer enquires about seat availability on a particular flight, the transaction then reads the no. of seats on that flight a second time before completing the reservation & it may end up showing a diffⁿ. value.

| $T_1$ | $T_2$ |
|---|---|
| R(X) | |
| | R(X) |
| | W(X) |
| R(X). | |

Both reads have diffⁿ. values of x though $T_1$ didn't updated X

# Problems due to concurrent Transactions

1) Lost Update (WW)
2) Unrepeatable Read (RW)
3) The incorrect summary. (due to opertn of aggregate fxns)
4) Dirty Read (WR)
   Temporary

## Lost Update Problem (some update is lost) (write)

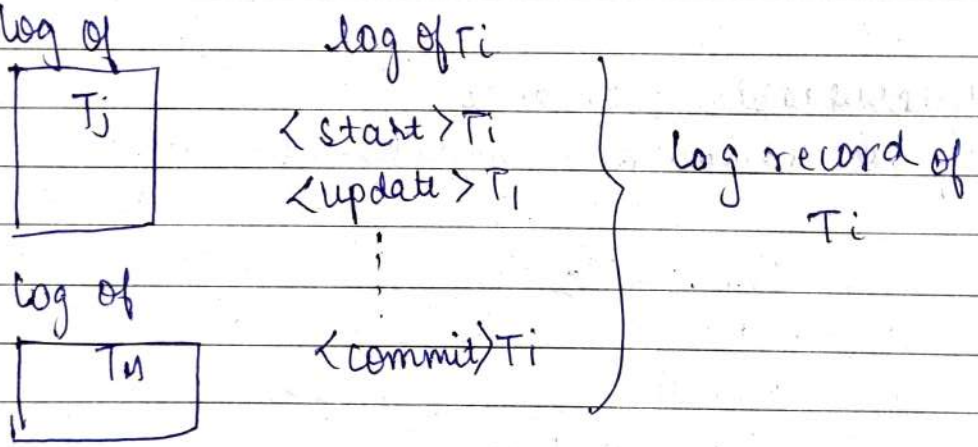
This problem occurs when 2 transactions that access the same database items have their opertn's interleaved in a way that makes the value of some database items incorrect

| | $T_1$ | $T_2$ |
|---|---|---|
| X:100<br>N=50<br>M=20. | R(X)<br>X:=X-N<br>// X=50 | |
| | | R(X)<br>X:=X+M // X:=120. |
| X=50 ← this value is lost | W(X) | |
| | R(Y) | |
| | Y:=Y+N<br>W(Y) | |
| | | W(X) → 120<br>↓<br>item X has an incorrect value. |

# Recoverability

- **Recoverable** → Change can be undo → Previous state can be restored
- **Irrecoverable** → Changes can't be undone → previous state can't be restored

Procedures: When a transaction (say $T_i$) executes, there is always a transaction log created corresponding to it.




Update Example

$\langle T_0, X_j, v_i, v_j \rangle$

$\langle T_0, A, 100, 200 \rangle$

When Roll backs previous value of Ti is stored.

After a commit, there wont be rollback it makes the schedule irrecoverable as the log is delete after schedule commits.

Schedule On the basis of Recoverability

1) Irrecoverable schedule
2) Cascading Rollback (Cascading Recoverable)
3) Strict Recoverability (Third level of recoverability)

Irrecoverable schedule
If we rollback a committed transaction

| $T_1$ | $T_2$ |
|---|---|
| R(A) | |
| W(A) | |
| | R(A) |
| | W(A) |
| | commit |
| W(A) | |
| Rollback | |

Not Possible

Cascading Rollback → It is a type of Rollback when, because of the Rollback of 1 transaction Rollback of many other transactions is caused.

| $T_1$ | $T_2$ | $T_3$ | $T_4$ |
|---|---|---|---|
| W(A) | | | |
| | R(A) | | |
| | W(A) | | |
| | | R(A) | |
| | | W(A) | |
| | | | R(A) |
| Rollback. | | | W(A) |

Rollback of $T_1$ causes Rollback of all $T_2$, $T_3$, $T_4$.

Soln → Transaction which completes first should commit first

| $T_1$ | $T_2$ | $T_3$ | $T_4$ |
|---|---|---|---|
| W(A) | | | |
| commit) | R(A) | | |
| | W(A) | | |
| | commit | | |
| | | R(A) | |
| | | W(A) | |
| | | commit | R(A) |
| | | | W(A) |
| | | | commit |

Disadvantage → unnecessary wastage of CPU time.

Problem of (WR) has been removed

Remaining problem

→ R W

→ W W.

Shortcut → To check if a scheduling is Cascading Recoverable.

1) Check for WR cond$^n$, If no WR problem exists, it is cascading Recoverable schedule.

3) Strict Recoverability.

The dependent transaction should read or write only after the commit of the Independent Transact $T_1$ here.

| Independent transaction → $T_1$ | $T_2$ ← Dependent Transaction |
|---|---|
| R(A) | |
| W(A) | |
| commit | |
| | R(A) / W(A) |

So $T_2$ show R/W only after $T_1$.

Advantages
→ WW also remove as well as WR.

→ R-W still left

Types of schedules

Recoverable — Irrecoverable

Cascading recoverable (level 0)
Cascadeless recoverable (level 1)
Strict recoverable (level 2)

Cascading Rollback → When Rollback of 1 transaction causes the Rollback of some other transactions, then the phenomena is called as Cascading Rollback.

Imp Note → All transactions haven't performed their rollback.

→ The phenomena of Cascading Rollback occurs in some recoverable schedules where an uncomitted transaction has to rolled back because it Read an item from a transaction that failed.

## Cascadeless schedule

A schedule is said to be cascadeless, if every transaction in the schedule reads only item that were written by committing transactions.

Ex.

| $T_1$ | $T_2$ | $T_3$ | $T_4$ |
|---|---|---|---|
| W(A) | | | |
| | R(A) | | |
| | W(A) | | |
| | | R(A) | |
| | | W(A) | |
| $T_1$ Aborts ← | | | R(A) |
| | | | W(A) |

| $T_1$ | $T_2$ |
|---|---|
| R(A) | |
| W(A) | |
| Commit / Rollback | |
| | R(A) |

The problem of (R(W) Dirty Read is removed.

Check if cascadeless Recoverable / Irrecoverable.

$R_1(x)$ $R_2(z)$ $R_1(z)$ $R_3(a)$ $R_3(y)$ $W_1(x)$ $W_3(y)$ $R_2(y)$ $W_2(z)$

($W_R$)

Soln

Step 1 → Checking if Recoverable / Irrecoverable Transactions are committing in the same order as they are completing their work.

Step 2 → Check if RW problem is there.

$W_3(y)$ $R_2(y)$, Therefore it is not cascadeless recoverable.

Strict schedule → cascadeless
Cascadeless schedule → recoverable.

Check for strict Recoverability

$R_1(x)$ $R_2(z)$ $R_3(x)$ $R_1(z)$ $R_2(y)$ $R_3(y)$ $W_1(x)$ $C_1$ $W_2(z)$ $W_3(y)$ $W_2(y)$ $C_3$ $C_2$

| 1) Level of completion | commit there |
|---|---|
| $C_1$ | $C_1$ |
| $C_3$ | $C_3$ |
| $C_2$ | $C_2$ |

Recoverable